Università degli Studi di Salerno

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione
Ciclo 30 - a.a. 2016/2017

Tesi di Dottorato / Ph.D. Thesis

Round and Computational Efficiency of Two-Party Protocols

Supervisor
Prof. Giuseppe PERSIANO

Candidate
Michele CIAMPI

Ph.D. Program Director
Prof. Pasquale CHIACCHIO

Dipartimento di Ingegneria dell'Informazione
ed Elettrica e Matematica Applicata
Dipartimento di Informatica

Abstract

A cryptographic protocol is defined by the behaviour of the involved parties and the messages that those parties send to each other. Beside the functionality and the security that a cryptographic protocol provides, it is also important that the protocol is *efficient*. In this thesis we focus on the efficiency parameters of a cryptographic protocol related to the computational and round complexity. That is, we are interested in the computational cost that the parties involved in the protocol have to pay and how many interactions between the parties are required to securely implement the functionality which we are interested in. Another important aspect of a cryptographic protocol is related to the computational assumptions required to prove that the protocol is secure. The aim of this thesis is to improve the state of the art with respect to some cryptographic functionalities where two parties are involved, by providing new techniques to construct more efficient cryptographic protocols whose security can be proven by relying on better cryptographic assumptions.

The thesis is divided in three parts. In the first part we consider *Secure Two-Party Computation* (2PC), a cryptographic technique that allows to compute a functionality in a secure way. More precisely, there are two parties, Alice and Bob, willing to compute the output of a function f given x and y as input. The values x and y represent the inputs of Alice and Bob respectively. Moreover, each party wants to keep the input secret while allowing the other party to correctly compute $f(x, y)$. As a first result, we show the first secure 2PC protocol with black box simulation, secure under standard and generic assumption, with optimal round complexity in the simultaneous message exchange model. In the simultaneous message exchange model both parties can send a message in each round; in the rest of this thesis we assume the in each round only one party can send a message.

We advance the state of the art in secure 2PC also in a relaxed setting. More precisely, in this setting a malicious party that attacks the protocol to understand the secret input of the honest party, is forced to follow the protocol description. Moreover, we consider the case in which the parties want to compute in a secure way the *Set-Membership* functionality. Such a functionality allows to check whether an element belongs to a set or not. The proposed protocol improves the state of the art both in terms of performance and generality. In the second part of the thesis we show the first 4-round *concurrent non-malleable commitment* under one-way functions. A commitment scheme allows the sender to send an encrypted message, called commitment, in such a way that the message inside the commitment cannot be opened until that an *opening* information is provided by the sender. Moreover, there is a unique way in which the commitment can be open. In this thesis we consider the case in which the sender sends the commitment (e.g. trough a computer network) that can be eavesdropped by an adversary. In this setting the adversary can catch the commitment C and modify it thus obtaining a new commitment C' that contains a message related to the content of C. A non-malleable commitment scheme prevents such attack, and our scheme can be proved secure even in the case that the adversary can eavesdrop multiple commitments and in turn, compute and send multiple commitments.

The last part of the thesis concerns *proof systems*. Let us consider an $\mathcal{NP}$-language, like the language of graph Hamiltonicity. A proof system allows an entity called *prover* to prove that a certain graph (instance) contains a Hamiltonian cycle (witness) to another entity called *verifier.* A proof system can be easily instantiated in one round by letting the prover to send the cycle to the verifier. What we actually want though, is a protocol in which the prover is able to convince the verifier that a certain graph belongs to the language of graph Hamiltonicity, but in such a way that no information about the cycle is leaked to the verifier. This kind of proof systems are called *Zero Knowledge.* In this thesis we show a non-interactive Zero-Knowledge proof system, under the assumption that both prover and verifier have access to some honestly generated *common reference string* (CRS). The provided construction improves the state of the art both in terms of efficiency and generality. We consider also the scenario in which prover and verifier do not have access to some honestly generated information and study the notion of *Witness Indistinguishability.* This notion considers instances that admit more than one witness, e.g. graphs that admit two distinct Hamiltonian cycle (as for the notion of Zero Knowledge, the notion of Witness Indistinguishability makes sense for all the languages in $\mathcal{NP}$, but for ease of exposition we keep focusing our attention of the language of graph Hamiltonicity). The security notion of Witness-Indistinguishability ensures that a verifier, upon receiving a proof from a prover, is not able to figure out which one of the two Hamiltonian cycles has been used by the prover to compute the proof. Even though the notion of Witness Indistinguishability is weaker than the notion of Zero Knowledge, Witness Indistinguishability is widely used in many cryptographic applications. Moreover, given that a Witness-Indistinguishable protocol can be constructed using just three rounds of communication compared to the four rounds required to obtain Zero Knowledge (with black-box simulation), the use of Zero-Knowledge as a building block to construct a protocol with an optimal number of rounds is sometimes prohibitive. Always in order to provide a good building block to construct more complicated cryptographic protocols with a nice round complexity, a useful property is the so called *Delayed-Input* property. This property allows the prover to compute all but the last round of the protocol without knowing the instance nor the witness. Also, the Delayed-Input property allows the verifier to interact with the prover without knowing the instance at all (i.e. the verifier needs the instance just to decide whether to accept or not the proof received by the prover). In this thesis we provide the first efficient Delayed-Input Witness-Indistinguishable proof system that consists of just three round of communication.

Acknowledgments

The first person that I want to thank is my advisor, Giuseppe Persiano. He introduced and guided me into the to world of cryptography and I am honored to have been advised by him. A huge thank you goes to Ivan Visconti, he taught me a lot and an important part of what I am now is thanks to him. He was, and still is, a source of good advice not only in research, but also in life. The dedication that Giuseppe and Ivan put into my formation is something that I would have never expected when I started my PhD and I will be always grateful to them for this.

I am grateful to Rafail Ostrovsky that hosted me at UCLA, and gave me the opportunity to spend part of my PhD in an incredible research group. My period in Los Angeles gave me a lot under many aspects, and the time I spent working with a person with the experience and the kindness of Rafail represents one of the most intense part of my PhD.

I am also thankful to Ivan Damgård and Claudio Orlandi for the warm welcome they gave me when I visited Aarhus University. The time I spent at Aarhus was pleasant, and I had the opportunity to work in an exciting environment with a research group full of nice and great people.

A massive thank you is mandatory for Luisa Siniscalchi. We started the journey of the PhD together and we have been working side by side in many occasions. She has always been patient with me (which is not so easy) and an authentic friend, always ready to help me.

I am deeply thankful to my parents: Giuseppe and Giuseppina. They taught me the meaning of working and implicitly pushed me toward the best I could achieve. They supported me in many aspects of my life, and their wisdom, strength and acuity represent a huge aspiration for me. I am thankful to my sister Caterina for all the advices she tried to gave. Her morality represented and still represents a lighthouse for the darkest moments.

Last but not the least I want to say thank you to Paola, I cannot immagine what these three years would have been without her.

Contents

Chapter 1

Introduction

The aim of a cryptographic protocol is, in general, to hide information from a malicious party that does not have the permission to access to such information. This thesis focuses on cryptographic protocols where two entities are involved. More precisely, we are going to consider three classes of cryptographic functionality, and advance the state of the art providing more efficient protocols to securely implements those functionality classes. In more details, in Part I we consider *Secure Two-Party Computation* (2PC), and provide the first secure 2PC protocol with black box-simulation, secure under standard and generic assumptions, with optimal round complexity in the simultaneous message exchange model as stated in [COSV17c]. We also show our new approach, proposed in [CO18], to securely implement the *Set Membership* functionality in the semi-honest setting. The proposed construction can be combined with the right 2PC techniques to achieve more efficient protocols for computations of the form $z = f(X \cap Y)$ for arbitrary functions f. In Part II we show a 4-round *concurrent non-malleable commitment* scheme under the one-way functions (OWFs) as stated in [COSV17b], and in the last part we study the proof systems[1]. More precisely, we first focus on the question of achieving *adaptive-input* proofs of partial knowledge, showing an efficient construction, as stated in [CPS$^+$16a], of a 3-round public-coin witness-indistinguishable (k, n)-proof of partial knowledge where *all* instances can be decided in the third round. For the latest contribution of Part III we consider the notion of non-interactive Zero-Knowledge proof and show the construction provided in [CPSV16], that improves the state of the art both in terms of efficiency and generality. We now give more details about the contributions provided in each part of this thesis.

1.1 Secure Two-Party Computation.

Obtaining round-optimal secure computation [Yao82, GMW87] has been a long standing open problem. For the two-party case the work of Katz and Ostrovsky [KO04] demonstrated that 5 rounds are both necessary and sufficient, with black-box simulation, when both parties need to obtain the output. Their construction relies on the use of trapdoor permutations[2]. A more recent work of Ostrovsky et al. [ORS15] showed that a black-box use of trapdoor permutations is sufficient for obtaining the above round-optimal construction.

A recent work of Garg et al. [GMPP16] revisited the lower bound of [KO04] when the communication channel allows both players to send messages in the same round, a setting that

[1]When discussing informally we will use the word proof to mean both an unconditionally sound proof and a computationally sound proof (i.e., an argument). Only in the formal part of the thesis we will make a distinction between arguments and proofs.

[2]The actual assumption is *enhanced* trapdoor permutations, but for simplicity in this work we will omit the word *enhanced* assuming it implicitly.

has been widely used when studying the round complexity of multi-party computation. Focusing on the simultaneous message exchange model, Garg et al. showed that 4 rounds are necessary to build a secure two-party computation (2PC) protocol for every functionality with black-box simulation. In the same work they also designed a 4-round secure 2PC protocol for every functionality. However their construction compared to the one of [KO04] relies on much stronger complexity assumptions. Indeed the security of their protocol crucially relies on the existence of a 3-round 3-robust [Pol16] parallel non-malleable commitment scheme. According to [Pol16] such commitment scheme can be constructed either through non-falsifiable assumptions (i.e., using the construction of [PPV08]) or through sub-exponentially-strong assumptions (i.e., using the construction of [COSV16]). In very recent works [HHPV17, BGJ$^+$17] it is showed how to construct a 4-round protocol to securely compute every functionality for the multi-party case under the Decisional Diffie-Hellman (DDH) assumption for the case of [HHPV17] and under the LWE+DDH assumptions for the case of [BGJ$^+$17].

Even given this new results, we have a gap in the state of affairs that leaves open the following interesting open question:
Open Question: *is there a 4-round construction for secure 2PC for any functionality in the simultaneous message exchange model assuming (standard) trapdoor permutations?*

In this thesis we answer positively to this question. Moreover, our construction for secure two-party computation relies on a special 4-round protocol for oblivious transfer that nicely composes with other protocols in parallel. We define and construct such special oblivious transfer protocol from trapdoor permutations. This building block is clearly interesting on its own. Our construction also makes use of a recent advance on non-malleability: a delayed-input 4-round non-malleable zero knowledge argument.

In this part of the thesis we also consider the semi-honest model and Private-Set Intersection (PSI), one of the most popular and practically relevant secure two-party computation tasks. Designing special-purpose PSI protocols (which are more efficient than generic 2PC solutions) is a very active line of research. In particular, a recent line of work has proposed PSI protocols based on oblivious transfer (OT) which, thanks to recent advances in OT-extension techniques, is nowadays a very cheap cryptographic building block. Unfortunately, these protocols cannot be plugged into larger 2PC applications since in these protocols one party (by design) learns the output of the intersection. Therefore, it is not possible to perform secure *post-processing* of the output of the PSI protocol. In this thesis we propose a novel and efficient OT-based PSI protocol that produces an "encrypted" output that can therefore be later used as an input to other 2PC protocols. In particular, the protocol can be used in combination with all common approaches to 2PC including garbled circuits, secret sharing and homomorphic encryption. Thus, our protocol can be combined with the right 2PC techniques to achieve more efficient protocols for computations of the form $z = f(X \cap Y)$ for arbitrary functions f.

1.2 4-Round Concurrent Non-Malleable Commitment from OWFs

Commitment schemes are a fundamental primitive in Cryptography. Here we consider the intriguing question of constructing round-efficient schemes that remain secure even against man-in-the-middle (MiM) attacks: non-malleable (NM) commitments [DDN91].

Non-malleable commitments. The round complexity of commitment schemes in the stand-alone setting is nowadays well understood. Non-interactive commitments can be constructed assuming the existence of 1-to-1 one-way functions (OWFs) [GL89]; 2-round commitments can be constructed assuming the existence of OWFs only. Moreover non-interactive commitments do not exist if one relies on the black-box use of OWFs only [MP12]. Instead, the round complexity of NM commitments after 25 years of research remains a fascinating open question, in particular when taking into account the required computational assumptions. The original construction of [DDN91] required a logarithmic number of rounds and the sole use of OWFs. Then, through a long sequence of very exciting positive results [Bar02, PR03, PR05b, PR05a, PR08b, PR08a, LPV08, PW10, Wee10, LP11, LP15, Goy11, GLOV12], the above open question has been in part solved obtaining a constant-round[3] (even concurrent) NM commitment scheme by using any OWF in a black-box fashion. On the negative side, Pass proved that NM commitments require at least 3 rounds [Pas13][4] when security is proved through a black-box reduction to polynomial-time hardness assumptions.

Breaking the multiple rewind-slot barrier. The above papers left open the question of achieving (concurrent) non-malleable commitments with optimal round complexity. A main common issue for round-efficient non-malleable commitments is that typically a security proof requires some simulation on the left and extraction on the right that should not interfere with each other. Indeed, a known paradigm introduced by Pass [Pas04] proposes to have in a protocol multiple potential rewind slots so that extraction and simulation can both be run without in 2 independent sequential steps. On the negative side, the use of multiple rewind slots increases the round complexity of the protocol (i.e., two slots require at least 5 rounds).

More recently the multiple rewind-slot technique has been bypassed in [GRRV14] but only for the (simpler) one-one case (i.e., just one sender and one receiver). In particular, Goyal et al. [GRRV14] showed a *one-one* 4-round NM commitment scheme based on OWFs only. The more recent work of Goyal et al. [GPR16] exploited the use of the NM codes in the split-state model of Aggarwal et al. [ADL14] to show a 3-round one-one NM commitment scheme based on the black-box use of any 1-to-1 OWF that is secure against super-polynomial time adversaries. Ciampi et al. [COSV16] obtained concurrent non-malleability in 3 rounds starting from any one-one non-malleable (and extractable) commitment scheme, but their security proof crucially relies on the existence of one-way permutations secure against subexponential-time adversaries. Assumptions against super-polynomial time adversaries allow to avoid multiple rewind slots even in presence of polynomially many sessions since the security proof can rely on straight-line simulation/extraction[5]. Recently, the work of Khurana [Khu17] appeared in TCC 2017, provides a 3-round non-malleable commitment relying on the DDH assumption.

In this work we break the multiple-slot barrier for concurrent NM commitments by showing a 4-round scheme based on the sole existence of OWFs. While previous work relied on having either 1) stronger assumptions or 2) multiple rewind slots or 3) non-generic assumptions, in this work we introduce new techniques that allow to have just one rewind slot, minimal hardness assumptions and full concurrency. More specifically we give the following four contributions.

Non-malleable commitments w.r.t. non-aborting adversaries. We prove that a subprotocol of [GRRV14] is a 4-round statistically binding concurrent NM commitment scheme from OWFs (resp. a 3-round perfectly binding concurrent NM commitment scheme from 1-to-1 OWFs), if the adversary is restricted to playing well-formed commitments in the right sessions when receiving well formed commitments from the left sessions. We refer to this weaker security notion as concurrent weak non-malleability (wNM).

Simulation-Witness-Independence. We define a new security notion for argument systems w.r.t. man-in-the-middle attacks that we refer to as simulation-witness-independence (SimWI). This security notion seemingly is not implied by previous notions as simulation-extractability/soundness and strong non-malleable witness indistinguishability.

[3]The construction of [GLOV12] can be compressed to 6 rounds (see [GRRV14]).

[4]If instead one relies on non-standard assumptions or trusted setups (e.g., using trusted parameters, working in the random oracle model, relying on the existence of NM OWFs) then there exist non-interactive NM commitments [DG03, PPV08].

[5]Hardness assumptions against subexponential-time adversaries were already used in [PR03, PW10, Wee10] to improve the round-complexity of NM commitments.

4-Round One-Many SimWI from OWFs. We then construct a 4-round one-many SimWI argument of knowledge for same specific languages by relying on OWFs only. This construction circumvents the major problem caused by the need of rewinding on the left to simulate and on the right to extract when there is only one available slot.

Concurrent wNM + One-Many SimWI $\Rightarrow$ 4-Round Concurrent NM Commitments. We present our new paradigm consisting in combining the above two notions in a protocol that runs in parallel the concurrent wNM commitment scheme and the one-many SimWI argument of knowledge. Therefore as main result of this work we upgrade concurrent wNM to full-fledged concurrent non-malleability without any penalization in rounds and assumptions.

1.3 Efficient and Delayed-Input Proof System

Proofs of partial knowledge allow a prover to prove knowledge of witnesses for k out of n instances of $\mathcal{NP}$ languages. Cramer, Damgård and Schoenmakers [CDS94] provided an efficient construction of a 3-round public-coin witness-indistinguishable (k, n)-proof of partial knowledge for any $\mathcal{NP}$ language, by cleverly combining n executions of Σ-protocols for that language. This transform assumes that all n instances are fully specified before the proof starts, and thus directly rules out the possibility of choosing some of the instances after the first round. In [CPS+16a] an improved transform where one of the instances can be specified in the last round is provided. The authors of [CPS+16a] focus on $(1, 2)$-proofs of partial knowledge with the additional feature that one instance is defined in the last round, and could be *adaptively* chosen by the verifier. They left as an open question the existence of an efficient $(1, 2)$-proof of partial knowledge where no instance is known in the first round. More in general, they left open the question of constructing an efficient (k, n)-proof of partial knowledge where knowledge of *all* n instances can be postponed. Indeed, this property is achieved only by inefficient constructions requiring $\mathcal{NP}$ reductions [LS90].

In this thesis we focus on the question of achieving *adaptive-input* proofs of partial knowledge. We provide through a transform the first efficient construction of a 3-round public-coin witness-indistinguishable (k, n)-proof of partial knowledge where *all* instances can be decided in the third round. Our construction enjoys *adaptive-input* witness indistinguishability. Additionally, the proof of knowledge property remains also if the adversarial prover selects instances adaptively at last round as long as our transform is applied to a proof of knowledge belonging to the widely used class of proofs of knowledge described in [Mau15, CD98]. Since knowledge of instances and witnesses is not needed before the last round, we have that the first round can be precomputed and in the online/offline setting our performance is similar to the one of [CDS94].

Our new transform relies on the DDH assumption (in contrast to the transforms of [CDS94, CPS+16a] that are unconditional). We also show how to strengthen the transform of [CPS+16a] so that it also achieves adaptive soundness, when the underlying combined protocols belong to the class of protocols described in [Mau15, CD98].

In the last part of this thesis we study the Fiat-Shamir (FS) transform. This transform uses a hash function to generate, without any further overhead, non-interactive zero-knowledge (NIZK) argument systems from constant-round public-coin honest-verifier zero-knowledge (public-coin HVZK) proof systems. In the proof of zero knowledge, the hash function is modeled as a *programmable* random oracle (PRO). In TCC 2015, Lindell embarked on the challenging task of obtaining a similar transform with improved heuristic security. Lindell showed that, for several interesting and practical languages, there exists an efficient transform in the *non-programmable* random oracle (NPRO) model that also uses a common reference string (CRS). A major contribution of Lindell's transform is that zero-knowledge is proved without random oracles and this is an important step towards achieving efficient NIZK arguments in the CRS model without random oracles. In this work, we analyze the efficiency and generality of Lindell's transform and notice a significant gap when compared with the FS transform. We then propose a new transform that aims at filling this gap. Indeed our transform is almost as efficient as the FS transform and can be applied to a broad class of public-coin HVZK proof systems. Our transform requires a CRS and an NPRO in the proof of soundness, similarly to Lindell's transform.

Chapter 2

Preliminaries

We denote the security parameter by λ and use "$||$" as concatenation operator (i.e., if a and b are two strings then by $a||b$ we denote the concatenation of a and b). For a finite set Q, $x \leftarrow Q$ denotes a sampling of x from Q with uniform distribution. We use the abbreviation PPT that stands for probabilistic polynomial time. We use $\mathsf{poly}(\cdot)$ to indicate a generic polynomial function.

A *polynomial-time relation* Rel (or *polynomial relation*, in short) is a subset of $\{0,1\}^* \times \{0,1\}^*$ such that membership of (x,w) in Rel can be decided in time polynomial in $|x|$. For $(x,w) \in \mathsf{Rel}$, we call x the *instance* and w a *witness* for x. For a polynomial-time relation Rel, we define the $\mathcal{NP}$-language L_{Rel} as $L_{\mathsf{Rel}} = \{x | \exists w : (x,w) \in \mathsf{Rel}\}$. Analogously, unless otherwise specified, for an $\mathcal{NP}$-language L we denote by $\mathsf{Rel_L}$ the corresponding polynomial-time relation (that is, $\mathsf{Rel_L}$ is such that $L = L_{\mathsf{Rel_L}}$). We denote by $\hat{L}$ the language that includes both L and all well formed instances that do not have a witness. Moreover we require that membership in $\hat{L}$ can be tested in polynomial time. We implicitly assume that a PPT algorithm that is supposed to receive an instance in $\hat{L}$ will abort immediately if the instance does not belong to $\hat{L}$.

Let A and B be two interactive probabilistic algorithms. We denote by $\langle A(\alpha), B(\beta)\rangle(\gamma)$ the distribution of B's output after running on private input β with A using private input α, both running on common input γ. Typically, one of the two algorithms receives 1^λ as input. A *transcript* of $\langle A(\alpha), B(\beta)\rangle(\gamma)$ consists of the messages exchanged during an execution where A receives a private input α, B receives a private input β and both A and B receive a common input γ. Moreover, we will refer to the *view* of A (resp. B) as the messages it received during the execution of $\langle A(\alpha), B(\beta)\rangle(\gamma)$, along with its randomness and its input. We say that a protocol (A, B) is public coin if B sends to A random bits only. When it is necessary to refer to the randomness r used by and algorithm A we use the following notation: $A(\cdot; r)$.

2.1 Standard Definitions

Definition 1 (Proof/argument system). *A pair of* PPT *interactive algorithms* $\Pi = (\mathcal{P}, \mathcal{V})$ *constitutes a* proof system *(resp., an* argument system*) for an* $\mathcal{NP}$*-language* L*, if the following conditions hold:*

Completeness: *For every* $x \in L$ *and* w *such that* $(x,w) \in \mathsf{Rel_L}$*, it holds that:*

$$\text{Prob}\left[\ \langle \mathcal{P}(w), \mathcal{V}\rangle(x) = 1\ \right] = 1.$$

Soundness: *For every interactive (resp.,* PPT *interactive) algorithm* $\mathcal{P}^\star$*, there exists a negligible function* ν *such that for every* $x \notin L$ *and every* z*:*

$$\text{Prob}\left[\ \langle \mathcal{P}^\star(z), \mathcal{V}\rangle(x) = 1\ \right] < \nu(|x|).$$

A proof/argument system $\Pi = (\mathcal{P}, \mathcal{V})$ for an $\mathcal{NP}$-language L, enjoys *delayed-input* completeness if $\mathcal{P}$ needs x and w only to compute the last round and $\mathcal{V}$ needs x only to compute the output. Before that, $\mathcal{P}$ and $\mathcal{V}$ run having as input only the size of x. The notion of delayed-input completeness was defined in [CPS+16a]. We say that the transcript τ of an execution $b = \langle \mathcal{P}(z), \mathcal{V}\rangle(x)$ is *accepting* if $b = 1$.

Definition 2 (Computational indistinguishability). *Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles, where X_λ's and Y_λ's are probability distribution over $\{0,1\}^l$, for same $l = \mathsf{poly}(\lambda)$. We say that $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are* computationally indistinguishable, *denoted $X \approx Y$, if for every* PPT *distinguisher $\mathcal{D}$ there exists a negligible function ν such that for sufficiently large $\lambda \in \mathbb{N}$,*

$$\left|\text{Prob}\left[\ t \leftarrow X_\lambda : \mathcal{D}(1^\lambda, t) = 1\ \right] - \text{Prob}\left[\ t \leftarrow Y_\lambda : \mathcal{D}(1^\lambda, t) = 1\ \right]\right| < \nu(\lambda).$$

We note that in the usual case where $|X_\lambda| = \Omega(\lambda)$ and λ can be derived from a sample of X_λ, it is possible to omit the auxiliary input 1^λ. In this work we also use the definition of *Statistical Indistinguishability*. This definition is the same as Definition 2 with the only difference that the distinguisher $\mathcal{D}$ is unbounded. In this case use $X \equiv_s Y$ to denote that two ensembles are statistically indistinguishable.

Definition 3 (Proof of Knowledge [Dam10]). *A pair $(\mathcal{P}, \mathcal{V})$ of PPT interactive machines is a* proof of knowledge with knowledge error $k(\cdot)$ *for polynomial-time relation* Rel *if the following properties hold:*

- Completeness. *For every $(x, w) \in \mathsf{Rel}$, it holds that*

$$\text{Prob}\left[\ \langle \mathcal{P}(w), \mathcal{V}\rangle(x) = 1\ \right] = 1.$$

- Knowledge Soundness: *there exists a probabilistic oracle machine* $\mathsf{Extract}$, *called the* extractor, *such that for every interactive machine $\mathcal{P}^\star$ and for every input x accepted by $\mathcal{V}$ when interacting with $\mathcal{P}^\star$ with probability $\epsilon(x) > k(x)$, $\mathsf{Extract}^{\mathcal{P}^\star}(x)$ outputs a witness w for x. Moreover, the expected number of steps performed by* $\mathsf{Extract}$ *is bounded by* $\texttt{poly}(|x|)/(\epsilon(x) - k(x))$.

In our security proofs we make use of the following observation. An interactive protocol Π that enjoys the property of completeness and PoK (AoK) with negligible soundness error is a proof (an argument) system. Indeed suppose by contradiction that is not. By the definition of PoK (AoK) it is possible to extract the witness for every theorem $x \in \{0,1\}^\lambda$ proved by $\mathcal{P}^\star$ with probability greater than negligible; contradiction.

We also consider the *adaptive-input* PoK/AoK property for all the protocols that enjoy delayed-input completeness. Adaptive-input PoK/AoK ensures that the PoK/AoK property still holds when a malicious prover can choose the statement adaptively at the last round. More details about these notions are provided in Chapter 6.

Definition 4 (Witness Indistinguishable (WI)). *An argument/proof system $\Pi = (\mathcal{P}, \mathcal{V})$, is* Witness Indistinguishable (WI) *for a relation* Rel *if, for every malicious* PPT *verifier $\mathcal{V}^\star$, there exists a negligible function ν such that for all x, w, w' such that $(x, w) \in \mathsf{Rel}$ and $(x, w') \in \mathsf{Rel}$ it holds that:*

$$\left|\text{Prob}\left[\ \langle \mathcal{P}(w), \mathcal{V}^\star\rangle(x) = 1\ \right] - \text{Prob}\left[\ \langle \mathcal{P}(w'), \mathcal{V}^\star\rangle(x) = 1\ \right]\right| < \nu(|x|).$$

The notion of a perfect *WI argument/proof system is obtained by requiring that $\nu(|x|) = 0$.*

2.2 Commitment Schemes

Definition 5 (Commitment Scheme)**.** *Given a security parameter 1^λ, a commitment scheme* $\mathsf{CS} = (\mathsf{Sen}, \mathsf{Rec})$ *is a two-phase protocol between two* PPT *interactive algorithms, a sender* Sen *and a receiver* Rec*. In the commitment phase* Sen *on input a message m interacts with* Rec *to produce a commitment* $\mathtt{com}$*, and the private output* $\mathtt{d}$ *of* Sen*.*

In the decommitment phase, Sen *sends to* Rec *a decommitment information* $(m, \mathtt{d})$ *such that* Rec *accepts m as the decommitment of* $\mathtt{com}$*.*

Formally, we say that $\mathsf{CS} = (\mathsf{Sen}, \mathsf{Rec})$ *is a perfectly binding commitment scheme if the following properties hold:*

Correctness:

- *Commitment phase. Let* $\mathtt{com}$ *be the commitment of the message m given as output of an execution of* $\mathsf{CS} = (\mathsf{Sen}, \mathsf{Rec})$ *where* Sen *runs on input a message m. Let* $\mathtt{d}$ *be the private output of* Sen *in this phase.*
- *Decommitment phase*[1]*.* Rec *on input m and* $\mathtt{d}$ *accepts m as decommitment of* $\mathtt{com}$*.*

Statistical (resp. Computational) Hiding([Lin10]): *for any adversary (resp.* PPT *adversary) $\mathcal{A}$ and a randomly chosen bit $b \in \{0, 1\}$, consider the following hiding experiment* $\mathsf{ExpHiding}^b_{\mathcal{A},\mathsf{CS}}(\lambda)$*:*

- *Upon input 1^λ, the adversary $\mathcal{A}$ outputs a pair of messages m_0, m_1 that are of the same length.*
- Sen *on input the message m_b interacts with $\mathcal{A}$ to produce a commitment of m_b.*
- *$\mathcal{A}$ outputs a bit b' and this is the output of the experiment.*

For any adversary (resp. PPT *adversary) $\mathcal{A}$, there exist a negligible function ν, s.t.:*

$$\left| \mathrm{Prob}\left[\mathsf{ExpHiding}^0_{\mathcal{A},\mathsf{CS}}(\lambda) = 1 \right] - \mathrm{Prob}\left[\mathsf{ExpHiding}^1_{\mathcal{A},\mathsf{CS}}(\lambda) = 1 \right] \right| < \nu(\lambda).$$

Statistical (resp. Computational) Binding: *for every commitment* $\mathtt{com}$ *generated during the commitment phase by a possibly malicious unbounded (resp. malicious* PPT*) sender* $\mathsf{Sen}^\star$ *there exists a negligible function ν such that* $\mathsf{Sen}^\star$*, with probability at most $\nu(\lambda)$, outputs two decommitments $(m_0, \mathtt{d}_0)$ and $(m_1, \mathtt{d}_1)$, with $m_0 \neq m_1$, such that* Rec *accepts both decommitments.*

We also say that a commitment scheme is perfectly binding *iff $\nu(\lambda) = 0$.*

When a commitment scheme $(\mathsf{Com}, \mathsf{Dec})$ *is non-interactive, to not overburden the notation, we use the following notation.*

– *Commitment phase.* $(\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{Com}(m)$ *denotes that* $\mathtt{com}$ *is the commitment of the message m and* $\mathtt{dec}$ *represents the corresponding decommitment information.*

– *Decommitment phase.* $\mathsf{Dec}(\mathtt{com}, \mathtt{dec}, m) = 1$.

2-Round Instance-Dependent Trapdoor Commitments. Following [COSV17b] here we define a special commitment scheme based on an $\mathcal{NP}$-language L where sender and receiver also receive as input an instance x. While correctness and computational hiding hold for any x, we require that statistical binding holds for $x \notin L$ and moreover knowledge of a witness for $x \in L$ allows to equivocate. Finally, we require that a commitment along with two valid openings to different messages allows to compute the witness for $x \in L$. We recall that $\hat{L}$ denotes the language that includes L and all well formed instances that are not in L.

[1]In this work we consider only non-interactive decommitment phase.

Definition 6 (2-Round Instance-Dependent Trapdoor Commitments)**.** *Let 1^λ be the security parameter, L be an $\mathcal{NP}$-language and $\mathsf{Rel_L}$ be the corresponding $\mathcal{NP}$-relation. A triple of* PPT *algorithms $\mathsf{TC} = (\mathsf{Sen}, \mathsf{Rec}, \mathsf{TFake})$ is a 2-Round Instance-Dependent Trapdoor Commitment scheme if the following properties hold.*

Correctness. *In the 1st round, Rec on input 1^λ and $x \in \hat{L}$ outputs ρ. In the 2nd round Sen on input the message m, 1^λ, ρ and $x \in L$ outputs $(\mathtt{com}, \mathtt{dec})$. We will refer to the pair $(\rho, \mathtt{com})$ as the commitment of m. Moreover we will refer to the execution of the above two rounds including the exchange of the corresponding two messages as the commitment phase. Then Rec on input m, x, $\mathtt{com}$, $\mathtt{dec}$ and the private coins used to generate ρ in the commitment phase outputs 1. We will refer to the execution of this last round including the exchange of $\mathtt{dec}$ as the decommitment phase. Notice that an adversarial sender $\mathsf{Sen}^\star$ could deviate from the behavior of Sen when computing and sending $\mathtt{com}$ and $\mathtt{dec}$ for an instance $x \in \hat{L}$. As a consequence Rec could output 0 in the decommitment phase. We will say that $\mathtt{dec}$ is a valid decommitment of $(\rho, \mathtt{com})$ to m for an instance $x \in \hat{L}$, if Rec outputs 1.*

Hiding. *Given a* PPT *adversary $\mathcal{A}$, consider the following hiding experiment $\mathsf{ExpHiding}^b_{\mathcal{A},\mathsf{TC}}(\lambda, x)$ for $b = 0, 1$ and $x \in \hat{L}_R$:*

- *On input 1^λ and x, $\mathcal{A}$ outputs a message m, along with ρ.*
- *The challenger on input x, m, ρ, b works as follows: if $b = 0$ then it runs Sen on input m, x and ρ, obtaining a pair $(\mathtt{com}, \mathtt{dec})$, otherwise it runs TFake on input x and ρ, obtaining a pair $(\mathtt{com}, \mathtt{aux})$. The challenger outputs $\mathtt{com}$.*
- *$\mathcal{A}$ on input $\mathtt{com}$ outputs a bit b' and this is the output of the experiment.*

We say that hiding *holds if for any* PPT *adversary $\mathcal{A}$ there exist a negligible function ν, s.t.:*

$$\left| \mathrm{Prob}\left[\ \mathsf{ExpHiding}^0_{\mathcal{A},\mathsf{TC}}(\lambda, x) = 1\ \right] - \mathrm{Prob}\left[\ \mathsf{ExpHiding}^1_{\mathcal{A},\mathsf{TC}}(\lambda, x) = 1\ \right] \right| < \nu(\lambda).$$

Special Binding. *There exists a* PPT *algorithm that on input a commitment $(\rho, \mathtt{com})$, the private coins used by Rec to compute ρ, and two valid decommitments $(\mathtt{dec}, \mathtt{dec}')$ of $(\rho, \mathtt{com})$ to two different messages m and m', outputs w s.t. $(x, w) \in \mathsf{Rel_L}$ with overwhelming probability.*

Instance-Dependent Binding. *For every malicious unbounded sender $\mathsf{Sen}^\star$ there exists a negligible function ν s.t. for a commitment $(\rho, \mathtt{com})$ $\mathsf{Sen}^\star$, with probability at most $\nu(\lambda)$, outputs two decommitments (m_0, d_0) and (m_1, d_1) with $m_0 \neq m_1$ s.t. Rec on input the private coins used to compute ρ and $x \notin L$ accepts both decommitments.*

Trapdoorness. *For any* PPT *adversary $\mathcal{A}$ there exist a negligible function ν, s.t. for all $x \in L$ it holds that:*

$$\left| \mathrm{Prob}\left[\ \mathsf{ExpCom}_{\mathcal{A},\mathsf{TC}}(\lambda, x) = 1\ \right] - \mathrm{Prob}\left[\ \mathsf{ExpTrapdoor}_{\mathcal{A},\mathsf{TC}}(\lambda, x) = 1\ \right] \right| < \nu(\lambda)$$

where $\mathsf{ExpCom}_{\mathcal{A},\mathsf{TC}}(\lambda, x)$ and $\mathsf{ExpTrapdoor}_{\mathcal{A},\mathsf{TC}}(\lambda, x)$ are defined below[2].

[2]We assume wlog that $\mathcal{A}$ is stateful.

$\mathsf{ExpCom}_{\mathcal{A},\mathsf{TC}}(\lambda, x)$*:*	$\mathsf{ExpTrapdoor}_{\mathcal{A},\mathsf{TC}}(\lambda, x)$*:*
-*On input* 1^λ *and* x*,* $\mathcal{A}$ *outputs* (ρ, m)*.*	-*On input* 1^λ *and* x*,* $\mathcal{A}$ *outputs* (ρ, m)*.*
-Sen *on input* 1^λ*,* x*,* m *and* ρ*, outputs* $(\mathtt{com}, \mathtt{dec})$*.*	-TFake *on input* 1^λ*,* x *and* ρ*, outputs* $(\mathtt{com}, \mathtt{aux})$*.*
	-TFake *on input* $\mathtt{tk}$ *s.t.* $(x, \mathtt{tk}) \in \mathsf{Rel_L}$*,* x*,* ρ*,* $\mathtt{com}$*,* $\mathtt{aux}$ *and* m *outputs* $\mathtt{dec}$*.*
-$\mathcal{A}$ *on input* $(\mathtt{com}, \mathtt{dec})$ *outputs a bit* b *and this is the output of the experiment.*	-$\mathcal{A}$ *on input* $(\mathtt{com}, \mathtt{dec})$ *outputs a bit* b *and this is the output of the experiment.*

In this work we consider also a non-interactive version of Instance-Dependent Trapdoor Commitments. The only difference in the definition is that the first round sent by the receiver to the sender just disappears. In this case we use the following simplified notation.

- *Commitment phase.* $(\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{Sen}(m, 1^\lambda, x)$ *denotes that* $\mathtt{com}$ *is the commitment of the message* m *and* $\mathtt{dec}$ *represents the corresponding decommitment information.*
- *Decommitment phase.* $1 \leftarrow \mathsf{Rec}(m, x, \mathtt{com}, \mathtt{dec})$*.*
- *Trapdoor algorithms.* $(\mathtt{com}, \mathtt{aux}) \leftarrow \mathsf{TFake}(1^\lambda, x)$*,* $\mathtt{dec} \leftarrow \mathsf{TFake}(\mathtt{tk}, x, \mathtt{com}, \mathtt{aux}, m)$ *with* $(x, \mathtt{tk}) \in \mathsf{Rel_L}$*.*

In the rest of the work, we say that the sender uses the *honest procedure* when he computes the commitment $\mathtt{com}$ of a message m along with the decommitment information $\mathtt{dec}$ running Sen. Instead, the sender uses *trapdoor procedure* when he computes $\mathtt{com}$ and $\mathtt{dec}$ running TFake.

OWFs $\Rightarrow$ 2-round instance-dependent trapdoor commitments for any $\mathcal{NP}$ language. Here we recall the construction $(\mathsf{Sen_H}, \mathsf{Rec_H}, \mathsf{TFake_H})$ of [FS89] for Hamiltonian graphs already considered in [ORSV13] that satisfies Def. 6 and requires OWFs only.

$\mathsf{Sen_H}$ and $\mathsf{Rec_H}$ run as follows.

- $\mathsf{Rec_H} \rightarrow \mathsf{Sen_H}$. $\mathsf{Rec_H}$ on input a graph G with n nodes computes and sends ρ to the sender, where ρ is the 1st round of a two-round statistically binding commitment scheme from OWFs of [Nao91].

- $\mathsf{Sen_H}$ on input a bit b, ρ and a graph G with n nodes works as follows. If $b = 0$ then $\mathsf{Sen_H}$ picks a random permutation π and computes and sends the 2nd round of the statistically binding commitment using ρ as 1st round and committing one-by-one to all bits of the adjacency matrix of $\pi(G)$. If instead $b = 1$, then $\mathsf{Sen_H}$ computes and sends the 2nd round of the statistically binding commitment, using ρ as 1st round and committing to all bits of the adjacency matrix of a a graph that consists of a random cycle H of n nodes. In both cases, $(\rho, \mathtt{com} = (\mathtt{com}_1, \dots, \mathtt{com}_{n^2}))$ corresponds to the commitment of b. In the 1st case $\mathtt{dec}$ corresponds to the randomness used by $\mathsf{Sen_H}$, while in the 2nd case $\mathtt{dec}$ corresponds to the decommitments of those n edges in the adjacency matrix that correspond to the cycle.

- $\mathsf{Rec_H}$ on input a bit b, a graph G with n nodes, $\mathtt{dec}$ and $\mathtt{com}$ works as follows. If $b = 0$ then $\mathsf{Rec_H}$ verifies that $\mathtt{com}$ is a commitment of the adjacency matrix of $\pi(G)$ where both π and the decommitments of the adjacency matrix are taken from $\mathtt{dec}$. If instead $b = 1$ then $\mathsf{Rec_H}$ verifies that the decommitted edges in $\mathtt{dec}$ correspond to a cycle that was committed in $(\rho, \mathtt{com})$.

- $\mathsf{TFake_H}$ runs $\mathsf{Sen_H}$ on input ρ, a graph G with n nodes and $b = 0$ therefore obtaining $(\mathtt{com}, \mathtt{dec})$. Then $\mathsf{TFake_H}$ on input 0 and a cycle in G outputs $\mathtt{dec}$. Instead on input 1 and a cycle in G, $\mathsf{TFake_H}$ outputs the decommitments of the edges committed in $(\rho, \texttt{com})$ corresponding to a cycle in $\pi(G)$ where π was the permutation selected to compute $\texttt{com}$.

It is easy to see that the above construction is a 2-round instance-dependent trapdoor commitment scheme from OWFs. While the construction can be used to commit to a bit, in the rest of the thesis we will use this construction to commit to strings by implicitly assuming that the above steps are repeated in parallel for each bit of the string. Moreover, note that since Hamiltonicity is an $\mathcal{NP}$-complete language, the above construction works for any $\mathcal{NP}$-language through $\mathcal{NP}$ reductions. For simplicity in the rest of the thesis we will omit the $\mathcal{NP}$ reduction therefore assuming that the above scheme works directly on a given $\mathcal{NP}$-language L. We also observe that if a non-interactive statistically binding commitment scheme is used in the above construction, then we obtain an instance-dependent trapdoor commitment where the commitment phase is non-interactive as well. We also recall that non-interactive statistically binding commitment scheme can be constructed from one-to-one OWFs.

Part I

Secure Two-Party Computation

Chapter 3

Round Optimal 2-Party Computation

3.1 Introduction

In this chapter we show a 4-round construction for secure 2PC for any functionality in the simultaneous message exchange model assuming (standard) trapdoor permutations. Moreover our construction only requires black-box simulation and is therefore round optimal given the lower bound showed in [GMPP16]. We now describe our approach.

Along with the lower bound that we have mentioned above, in [GMPP16] is also provided a construction for 2PC that needs a 3-round 3-robust parallel non-malleable commitment, and constructing this primitive from standard polynomial-time assumptions is still an open problem. We circumvent the use of this primitive through a different approach. As done in [GMPP16], we start considering the 4-round 2PC protocol of [KO04] (KO protocol) that works only for those functionalities where only one player receives the output (we recall that the KO protocols do not assume the existence of a simultaneous message exchange channel). Then, as in [GMPP16] we consider two simultaneous executions of the KO protocol in order to make both parties able to obtain the output *assuming the existence of a simultaneous message exchange channel.* We describe now the KO protocol and then we explain how we manage to avoid 3-round 3-robust parallel non-malleable commitments.

The 4-round KO protocol. Following Fig. 3.1, at a very high level the KO protocol between the players P_1 and P_2, where only P_1 gets the output, works as follows. Let f be the function that P_1 and P_2 want to compute. In the second round P_2 generates, using his input, a Yao's garbled circuit C for the function f with the associated labels L. Then P_2 commits to C using a commitment scheme that is binding if P_2 runs the honest committer procedure. This commitment scheme however admits also an indistinguishable equivocal commitment procedure that allows later to open the equivocal commitment as any message. Let $\mathtt{com}_0$ be such commitment. In addition P_2 commits to L using a statistically binding commitment scheme. Let $\mathtt{com}_1$ be such commitment. In the last round P_2 sends the opening of the equivocal commitment to the message C. Furthermore, using L as input, P_2 in the 2nd and in the 4th round runs as a sender of a specific 4-round oblivious transfer protocol KOOT that is secure against a malicious receiver and secure against a semi-honest sender. Finally, in parallel with KOOT, P_2 computes a specific delayed-input zero-knowledge argument of knowledge (ZKAoK) to prove that the labels L committed in $\mathtt{com}_1$ correspond to the ones used in KOOT, and that $\mathtt{com}_0$ is binding since it has been been computed running the honest committer on input some randomness and some message. P_1 plays as a receiver of KOOT in order to obtain the labels associated to his input and computes the output of the two-party computation by running C on input the received labels. Moreover P_1 acts as a verifier for the ZKAoK where P_2 acts as a prover.

The 4-round protocol of Garg et al. In order to allow both parties to get the output in 4 rounds using a simultaneous message exchange channel, [GMPP16] first considers two simultaneous execution of the KO protocol (Fig. 3.2). Such natural approach yields to the following two problems (as stated in [GMPP16]): 1) nothing prevents an adversary from using two different inputs in the two executions of the KO protocol; 2) an adversary could adapt his input based on the input of the other party, for instance the adversary could simply forward the messages that he receives from the honest party. To address the first problem the authors of [GMPP16] add another statement to the ZKAoK where the player P_j (with $j = 1, 2$) proves that both executions of the KO protocol use the same input. The second problem is solved in [GMPP16] by using a 3-round 3-robust non-malleable commitment to construct KOOT and the ZKAoK in such a way that the input used by the honest party in KOOT cannot be mauled by the malicious party. The 3-robustness is required to avoid rewinding issues in the security proof. Indeed, in parallel with the 3-round 3-robust non-malleable commitment a WIPoK is executed in KOOT. At some point the security proof of [GMPP16] needs to rely on the witness-indistinguishability property of the WIPoK while the simulator of the ZKAoK is run. The simulator for the ZKAoK rewinds the adversary from the third to the second round, therefore rewinding also the challenger of the WIPoK of the reduction. To solve this problem [GMPP16, Pol16] rely on the stronger security of a 3-round 3-robust parallel non-malleable commitment scheme. Unfortunately, constructing this tool with standard polynomial-time assumptions is still an open question.

Our 4-round protocol. In our approach (that is summarized in Fig. 3.3), in order to solve problems 1 and 2 listed above using standard polynomial-time assumption (trapdoor permutations), we replace the ZKAoK and KOOT (that uses the 3-round 3-robust parallel commitment scheme) with the following two tools. 1) A 4-round delayed-input non-malleable zero-knowledge (NMZK) argument of knowledge (AoK) NMZK from one-way functions (OWFs) recently constructed in [COSV17a] (the theorem proved by NMZK is roughly the same as the theorem proved ZKAoK of [GMPP16]). 2) A new special OT protocol $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ that is *one-sided* simulatable [ORS15]. In this security notion for OT it is not required the existence of a simulator against a malicious sender, but only that a malicious sender cannot distinguish whether the honest receiver uses his real input or a fixed input (e.g., a string of 0s). Moreover some security against a malicious sender still holds even if the adversary can perform a mild form of "rewinds" against the receiver, and the security against a malicious receiver holds even when an interactive primitive (like a WIPoK) is run in parallel (more details about the security provided by $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ will be provided later).

Our security proof. In our security proof we exploit immediately the major differences with [GMPP16]. Indeed we start the security proof with an hybrid experiment where the simulator of NMZK is used, and we are guaranteed that the malicious party is behaving honestly by the non-malleability/extractability of NMZK. In the next hybrid experiment we use the simulator of the OT protocol $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ thus extracting the input from the adversary. In the rest of the hybrid experiments we remove the input of the honest party, and use the input extracted via $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ to complete the interaction against the adversary. An important difference with the approach used in [GMPP16] is that in all the steps of our security proof the simulator-extractor of NMZK is used to check every time that the adversary is using the same input in both the executions of the KO protocol even though the adversary is receiving a simulated NMZK of a false statement. More precisely, every time that we change something obtaining a new hybrid experiment, we prove that: 1) the output distributions of the experiments are indistinguishable; 2) the malicious party is behaving honestly (the statement proved by the NMZK given by the adversary is true). We will show that if one of these two invariants does not hold then we can make a reduction that breaks a cryptographic primitive.

The need of a special 4-round OT protocol. Interestingly, the security proof has to address a major issue. After we switch to the simulator of the NMZK, we have that in some hybrid experiment H_i, we need change the input of the receiver of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ (following the approach used in the security proof of the KO protocol). To demonstrate the indistinguishability between H_i and H_{i-1} we want to rely on the security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ against a malicious sender. Therefore we construct an adversarial sender $\mathcal{A}_{\mathcal{OT}}$ of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. $\mathcal{A}_{\mathcal{OT}}$ acts as a proxy for the messages of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ and internally computes the other messages of our protocol. In particular, the 1st and the 3rd rounds of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ are given by the challenger (that acts as a receiver of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$), and the 2nd and the 4th messages of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ are given by the malicious party. Furthermore, in order to compute the other messages of our 2PC protocol $\mathcal{A}_{\mathcal{OT}}$ needs to run the simulator-extractor of NMZK that, and this requires to rewind from the 3rd to 2nd round. This means that $\mathcal{A}_{\mathcal{OT}}$ needs to complete a 3rd round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$, for every different 2nd round that he receives (this is due to the rewinds made by the simulator of NMZK that are emulated by $\mathcal{A}_{\mathcal{OT}}$). We observe that since the challenger cannot be rewound, $\mathcal{A}_{\mathcal{OT}}$ needs a strategy to answer to these multiple queries w.r.t. $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ without knowing the randomness and the input used by the challenger so far. For these reasons we need $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ to enjoy an additional property: the *replayability* of the 3rd round. More precisely, given the messages computed by an honest receiver, the third round can be indistinguishability used to answer to any second round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ sent by a malicious sender. Another issue is that the idea of the security proof explained so far relies on the simulator-extractor of NMZK and this simulator rewinds also from the 4th to the 3rd round. The rewinds made by the simulator-extractor allow a malicious receiver to ask for different 3rd rounds of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. Therefore we need our $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ to be also secure against a more powerful malicious receiver that can send multiple (up to a polynomial γ) third rounds to the honest sender. As far as we know the literature does not provide an OT with the properties that we require, so in this work we also provide an OT protocol with these additional features. This clearly is of independent interest.

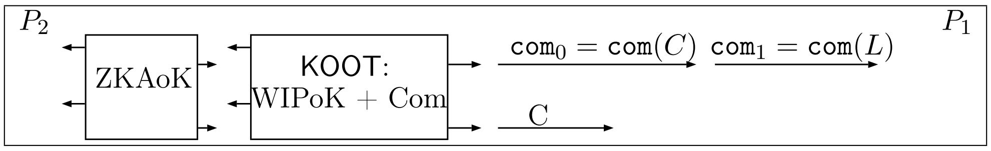

P_2
ZKAoK
KOOT: WIPoK + Com
$\texttt{com}_0 = \texttt{com}(C)$
$\texttt{com}_1 = \texttt{com}(L)$
C
P_1

Figure 3.1: The 4-round KO protocol from trapdoor permutations for functionalities where only one player receives the output.

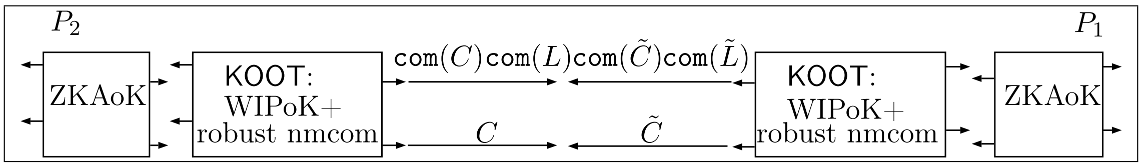

P_2
ZKAoK
KOOT: WIPoK+ robust nmcom
$\texttt{com}(C)\texttt{com}(L)\texttt{com}(\tilde{C})\texttt{com}(\tilde{L})$
C
$\tilde{C}$
KOOT: WIPoK+ robust nmcom
ZKAoK
P_1

Figure 3.2: The 4-round protocol of [GMPP16] for any functionality assuming 3-round 3-robust parallel non-malleable commitments in the simultaneous message exchange model.

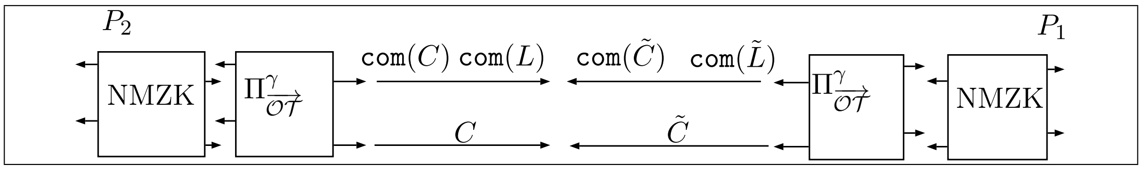

P_2
P_1
NMZK
$\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$
$\mathtt{com}(C)$ $\mathtt{com}(L)$
$\mathtt{com}(\tilde{C})$ $\mathtt{com}(\tilde{L})$
C
$\tilde{C}$
$\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$
NMZK

Figure 3.3: Our 4-round protocol for any functionality assuming trapdoor permutations in the simultaneous message exchange model.

3.2 Special One-Sided Simulatable OT

One of the main building blocks of our 2PC protocol is an OT protocol $\Pi^{\gamma}_{\mathcal{OT}} = (S_{\mathcal{OT}}, R_{\mathcal{OT}})$ one-sided simulatable[1]. Our $\Pi^{\gamma}_{\mathcal{OT}}$ has four rounds where the first (ot_1) and the third (ot_3) rounds are played by the receiver, and the remaining rounds (ot_2 and ot_4) are played by the sender. In addition $\Pi^{\gamma}_{\mathcal{OT}}$ enjoys the following two additional properties.

1. *Replayable third round.* Let $(\mathsf{ot}_1, \mathsf{ot}_2, \mathsf{ot}_3, \mathsf{ot}_4)$ be the messages exchanged by an honest receiver and a malicious sender during an execution of $\Pi^{\gamma}_{\mathcal{OT}}$. For any honestly computed ot'_2, we have that $(\mathsf{ot}_1, \mathsf{ot}_2, \mathsf{ot}_3)$ and $(\mathsf{ot}_1, \mathsf{ot}'_2, \mathsf{ot}_3)$ are identically distributed. Roughly, we are requiring that the third round can be reused in order to answer to any second round ot'_2 sent by a malicious sender.
2. *Repeatability.* We require $\Pi^{\gamma}_{\mathcal{OT}}$ to be secure against a malicious receiver $R^\star$ even when the last two rounds of $\Pi^{\gamma}_{\mathcal{OT}}$ can be repeated multiple times. More precisely a 4-round OT protocol that is secure in this setting can be seen as an OT protocol of $2 + 2\gamma$ rounds, with $\gamma \in \{1, \dots, \mathsf{poly}(\lambda)\}$ where λ represents the security parameter. In this protocol $R^\star$, upon receiving the 4th round, can continue the execution with $S_{\mathcal{OT}}$ by sending a freshly generated third round of $\Pi^{\gamma}_{\mathcal{OT}}$ up to total of γ 3rd rounds.
   Roughly, we require that the output of such $R^\star$ that runs $\Pi^{\gamma}_{\mathcal{OT}}$ against an honest sender can be simulated by an efficient simulator Sim that has only access to the ideal world functionality $F_{\mathcal{OT}}$ and oracle access to $R^\star$.

The security of $\Pi^{\gamma}_{\mathcal{OT}}$ is based on the existence of trapdoor permutations[2].

Our techniques. In order to construct $\Pi^{\gamma}_{\mathcal{OT}}$ we use as a starting point the following basic 3-round semi-honest OT Π_{sh} based on trapdoor permutations (TDPs) of [EGL82, KO04]. Let $l_0, l_1 \in \{0,1\}^\lambda$ be the input of the sender S and b be the input bit of the receiver R.

1. The sender S chooses a trapdoor permutation $(f, f^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends f to the receiver R.
2. R chooses $x \leftarrow \{0,1\}^\lambda$ and $z_{1-b} \leftarrow \{0,1\}^\lambda$, computes $z_b = f(x)$ and sends (z_0, z_1).
3. For $c = 0, 1$ S computes and sends $w_c = l_c \oplus \mathsf{hc}(f^{-1}(z_c))$

where $\mathsf{hc}(\cdot)$ is a hardcore bit of f. If the parties follow the protocol (i.e. in the semi-honest setting) then S cannot learn the receiver's input (the bit b) as both z_0 and z_1 are random strings. Also, due to the security of the TDP f, R cannot distinguish w_{1-b} from random as long as z_{1-b} is randomly chosen. If we consider a fully malicious receiver $R^\star$ then this protocol is not secure

[1]In the 2PC protocol we will actually use $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ that roughly corresponds to parallel executions of $\Pi^{\gamma}_{\mathcal{OT}}$. More details will be provided later.

[2]As suggested by Ivan Damgård and Claudio Orlandi in a personal communication, following the approach of [GKM+00], $\Pi^{\gamma}_{\mathcal{OT}}$ can be also constructed by relying on public key encryption schemes with special properties. More precisely the public key encryption scheme has to be such that that either the ciphertexts can be sampled without knowing the plaintext, or the public key can be sampled without knowing the corresponding secret key. In this work we give a formal construction and proof only for trapdoor permutations.

anymore. Indeed $R^\star$ could just compute $z_{1-b} = f(y)$ picking a random $y \leftarrow \{0,1\}^\lambda$. In this way $R^\star$ can retrieve both the inputs of the sender l_0 and l_1. In [KO04] the authors solve this problem by having the parties engaging a coin-flipping protocol such that the receiver is forced to set at least one between z_0 and z_1 to a random string. This is done by forcing the receiver to commit to two strings (r_0, r_1) in the first round (for the coin-flipping) and providing a witness-indistinguishable proof of knowledge (WIPoK) that either $z_0 = r_0 \oplus r'_0$ or $z_1 = r_1 \oplus r'_1$ where r'_0 and r'_1 are random strings sent by the sender in the second round. The resulting protocol, as observed in [ORS15], leaks no information to S about R's input. Moreover the soundness of the WIPoK forces a malicious $R^\star$ to behave honestly, and the PoK allows to extract the input from the adversary in the simulation. Therefore the protocol constructed in [KO04] is one-sided simulatable. Unfortunately this approach is not sufficient to have an OT protocol that has a *replayable* third round. This is due to the to the added WIPoK. More precisely, the receiver has to execute a WIPoK (acting as a prover) in the first three rounds. Clearly, there is no 3-round WIPoK such that given an accepting transcript (a, c, z) one can efficiently compute multiple accepting transcripts w.r.t. different second rounds without knowing the randomness used to compute a. This is the reason why we need to use a different approach in order to construct an OT protocol simulation-based secure against a malicious receiver that also has a replayable 3rd round.

Our construction: $\Pi^\gamma_{\mathcal{OT}}$. We start by considering a trick proposed in [ORS15]. In [ORS15] the authors construct a 4-round black-box OT starting from Π_{sh}. In order to force the receiver to compute a random z_{b-1}, in the first round R sends two commitments c_0 and c_1 such that $c_b = \mathsf{Eqcom}(\cdot), c_{1-b} = \mathsf{Eqcom}(r_{1-b})$. Eqcom is a commitment scheme that is binding if the committer runs the honest committer procedure; however this commitment scheme admits also an indistinguishable equivocal commitment procedure that allows later to open the equivocal commitment as any message. R then proves using a special WIPoK that either c_0 or c_1 is computed using the honest procedure (i.e., at least one of these commitments is binding). Then S in the second round computes $r'_0 \leftarrow \{0,1\}^\lambda$, $r'_1 \leftarrow \{0,1\}^\lambda$ and two TDPs f_0, f_1 with the respective trapdoor and sends (r'_0, r'_1, f_0, f_1) to R. R, upon receiving (r'_0, r'_1, f_0, f_1), picks $x \leftarrow \{0,1\}^\lambda$, computes $r_b = f_b(x) \oplus r'_b$ and sends the opening of c_{1-b} to the message r_{1-b} and the opening of c_b to the message r_b. At this point the sender computes and sends $w_0 = l_0 \oplus \mathsf{hc}(f_0^{-1}(r_0 \oplus r'_0))$, $w_1 = l_1 \oplus \mathsf{hc}(f_1^{-1}(r_1 \oplus r'_1))$. Since at least one between c_0 and c_1 is binding (due to the WIPoK), a malicious receiver can retrieve only one of the sender's input l_b. We observe that this OT protocol is still not sufficient for our propose due to the WIPoK used by the receiver (i.e., the 3rd round is not *replayable*). Moreover we cannot remove the WIPoK otherwise a malicious receiver could compute both c_0 and c_1 using the equivocal procedure thus obtaining l_0 and l_1. Our solution is to replace the WIPoK with some primitives that make replayable the 3rd round, still allowing the receiver to prove that at least one of the commitments sent in the first round is binding. Our key-idea is two use a combination of instance-dependent trapdoor commitment (IDTCom) and non-interactive commitment schemes. An IDTCom is defined over an instance x that could belong to the $\mathcal{NP}$-language L or not. If $x \notin L$ then the IDTCom is perfectly binding, otherwise it is equivocal and the trapdoor information is represented by the witness w for x. Our protocol is described as follows. R sends an IDTCom $\mathtt{tcom}_0$ of r_0 and an IDTCom $\mathtt{tcom}_1$ of r_1. In both cases the instance used is $\mathtt{com}$, a perfectly binding commitment of the bit b. The $\mathcal{NP}$-language used to compute $\mathtt{tcom}_0$ consists of all valid perfectly binding commitments of the message 0, while the $\mathcal{NP}$-language used to compute $\mathtt{tcom}_1$ consists of all valid perfectly binding commitments of the message 1.

This means that $\mathtt{tcom}_b$ can be opened to any value[3] and $\mathtt{tcom}_{1-b}$ is perfectly binding (we recall

[3]The decommitment information of $\mathtt{com}$ represents the trapdoor of the IDTCom $\mathtt{tcom}_b$.

that b is the input of the receiver). It is important to observe that due to the binding property of `com` it could be that both $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ are binding, but it can never happen that they are both equivocal. Now we can replace the two commitments and the WIPoK used in [ORS15] with $\mathtt{tcom}_0, \mathtt{tcom}_1$ and $\mathtt{com}(b)$ that are sent in the first round. The rest of the protocol stay the same as in [ORS15] with the difference that in the third round the openings to the messages r_0 and r_1 are w.r.t. $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$. What remains to observe is that when a receiver provides a valid third round of this protocol then the same message can be used to answer all second rounds. Indeed, a well formed third round is accepting if and only if the opening w.r.t. $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ are well computed. Therefore whether the third round is accepting or not does not depend on the second round sent by the sender.

Intuitively this protocol is also already secure when we consider a malicious receiver that can send multiple third rounds up to a total of γ 3rd rounds, thus obtaining an OT protocol of $2+2\gamma$ rounds (repeatability). This is because, even though a malicious receiver obtains multiple fourth rounds in response to multiple third rounds sent by $R^\star$, no information about the input of the sender is leaked. Indeed, in our $\Pi^{\gamma}_{\mathcal{OT}}$, the input of the receiver is fixed in the first round (only one between $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ can be equivocal). Therefore the security of the TDP ensures that only l_b can be obtained by $R^\star$ independently of what he does in the third round. In the formal part of the thesis we will show that the security of the TDP is enough to deal with such scenario.

We finally point out that the OT protocol that we need has to allow parties to use strings instead of bits as input. More precisely the sender's input is represented by $(l_0^1, l_1^1, \dots, l_0^m, l_1^m)$ where each l_b^i is an λ-bit length string (for $i = 1, \dots, m$ and $b = 0, 1$), while the input of the receiver is λ-bit length string.

This is achieved in two steps. First we construct an OT protocol where the sender's input is represented by just two m-bit strings l_0 and l_1 and the receiver's input is still a bit. We obtain this protocol by just using in $\Pi^{\gamma}_{\mathcal{OT}}$ a vector of m hard-core bits instead of just a single hard core bit following the approach of [KO04, GMPP16]. Then we consider m parallel execution of this modified $\Pi^{\gamma}_{\mathcal{OT}}$ (where the the sender uses a pair of strings as input) thus obtaining $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$.

3.3 Definitions and Tools

Definition 7 (Yao's garbled circuit). *We view Yao's garbled circuit scheme as a tuple of* PPT *algorithms* $(\mathsf{GenGC}, \mathsf{EvalGC})$ *where* GenGC *is the generation procedure which generates a garbled circuit for a circuit* $\mathsf{GC_y}$ *along with labels, and* EvalGC *is the evaluation procedure which evaluates the circuit on the correct labels. Each individual wire i of the circuit is assigned two labels, namely* $Z_{i,0}, Z_{i,1}$. *More specifically, the two algorithms have the following format:*

- $(Z_{1,0}, Z_{1,1}, \dots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC_y}) \leftarrow \mathsf{GenGC}(1^\lambda, F, y)$*:* GenGC *takes as input a security parameter λ, a circuit F and a string $y \in \{0,1\}^\lambda$. It outputs a garbled circuit* $\mathsf{GC_y}$ *along with the set of all input-wire labels* $\{Z_{1,b}, \dots, Z_{\lambda,b}\}_{b\in\{0,1\}}$. *The garbled circuit may be viewed as representing the function $F(\cdot, y)$.*

- $v = \mathsf{EvalGC}(\mathsf{GC_y}, Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda})$*: Given a garbled circuit* $\mathsf{GC_y}$ *and a set of input-wire labels Z_{i,x_i} where $x_i \in \{0,1\}$ for $i = 1, \dots, \lambda$,* EvalGC *outputs either an invalid symbol $\perp$, or a value $v = F(x, y)$.*

The following properties are required.

Correctness. $\mathrm{Prob}\,[\, F(x,y) = \mathsf{EvalGC}(\mathsf{GC_y}, Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda})\,] = 1$.

Security. *There exists a* PPT *simulator* SimGC *such that for any (F, x) and uniformly* *random labels $Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda}$, it holds that:*

$$(\mathsf{GC_y}, Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda}) \approx \mathsf{SimGC}(1^\lambda, F, x, v)$$

where $(Z_{1,0}, Z_{1,1}, \dots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC_y}) \leftarrow \mathsf{GenGC}(1^\lambda, F, y)$ and $v = F(x, y)$.

Definition 8 (Trapdoor permutation)**.** *Let $\mathcal{F}$ be a triple of* PPT *algorithms* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ *such that if $\mathsf{Gen}(1^\lambda)$ outputs a pair $(f, \mathtt{td})$, then $\mathsf{Eval}(f, \cdot)$ is a permutation over $\{0,1\}^\lambda$ and Invert $(f, \mathtt{td}, \cdot)$ is its inverse. $\mathcal{F}$ is a trapdoor permutation such that for all* PPT *adversaries $\mathcal{A}$:*

$$\text{Prob}\left[\ (f, \mathtt{td}) \leftarrow \mathsf{Gen}(1^\lambda); y \leftarrow \{0,1\}^\lambda, x \leftarrow \mathcal{A}(f, y) : \mathsf{Eval}(f, x) = y\ \right] \leq \nu(\lambda).$$

For convenience, we drop $(f, \mathtt{td})$ from the notation, and write $f(\cdot)$, $f^{-1}(\cdot)$ to denote algorithms $\mathsf{Eval}(f, \cdot)$, $\mathsf{Invert}(f, \mathtt{td}, \cdot)$ respectively, when f, $\mathtt{td}$ are clear from the context. Following [KO04, GMPP16] we assume that $\mathcal{F}$ satisfies (a weak variant of) "certifiability": namely, given some f it is possible to decide in polynomial time whether $\mathsf{Eval}(f, \cdot)$ is a permutation over $\{0,1\}^\lambda$. Let hc be the hardcore bit function for λ bits for the family $\mathcal{F}$. λ hardcore bits are obtained from a single-bit hardcore function h and $f \in \mathcal{F}$ as follows: $\mathsf{hc}(z) = h(z)||h(f(z))||\dots||h(f^{\lambda-1}(z))$. Informally, $\mathsf{hc}(z)$ looks pseudorandom given $f^\lambda(z)$[4].

3.3.1 Delayed-Input Non-Malleable Zero Knowledge

Here we follow [COSV17a]. The definition of [COSV17a] allows the adversary to explicitly select the statement, and as such the adversary provides also the witness for the prover. The simulated game however will filter out the witness so that the simulator will receive only the instance. This approach strictly follows the one of [SCO+01] where adaptive-input selection is explicitly allowed and managed in a similar way. As final remark, this definition will require the existence of a black-box simulator since a non-black-box simulator could retrieve from the code of the adversary the witness for the adaptively generated statement. The non-black-box simulator could then run the honest prover procedure, therefore canceling completely the security flavor of the simulation paradigm.

Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a delayed-input interactive argument system for a $\mathcal{NP}$-language L with witness relation $\mathsf{Rel_L}$. Consider a PPT MiM adversary $\mathcal{A}$ that is simultaneously participating in one left session and $\mathsf{poly}(\lambda)$ right sessions. Before the execution starts, $\mathcal{P}, \mathcal{V}$ and $\mathcal{A}$ receive as a common input the security parameter in unary 1^λ. Additionally $\mathcal{A}$ receives as auxiliary input $z \in \{0,1\}^\star$. In the left session $\mathcal{A}$ verifies the validity of the prove given by $\mathcal{P}$ with respect to the statement x (chosen adaptively in the last round of Π). In the right sessions $\mathcal{A}$ proves the validity of the statements $\tilde{x}_1, \dots, \tilde{x}_{\mathsf{poly}(\lambda)}$[5] (chosen adaptively in the last round of Π) to the honest verifiers $\mathcal{V}_1, \dots, \mathcal{V}_{\mathsf{poly}(\lambda)}$.

More precisely in the left session $\mathcal{A}$, before the last round of Π is executed, adaptively selects the statement x to be proved and the witness w, s.t. $(x, w) \in \mathsf{Rel_L}$, and sends them to $\mathcal{P}$.

Let $\mathsf{View}^{\mathcal{A}}(1^\lambda, z)$ denote a random variable that describes the view of $\mathcal{A}$ in the above experiment.

Definition 9 (Delayed-input NMZK)**.** *A delayed-input argument system $\Pi = (\mathcal{P}, \mathcal{V})$ for an $\mathcal{NP}$-language L with witness relation $\mathsf{Rel_L}$ is delayed-input non-malleable zero knowledge (NMZK) if for any MiM adversary $\mathcal{A}$ that participates in one left session and $\mathsf{poly}(\lambda)$ right sessions, there exists a expected* PPT *machine $S(1^\lambda, z)$ such that:*

[4] $f^\lambda(z)$ means the λ-th iteration of applying f on z.

[5] We denote (here and in the rest of the thesis) by $\tilde{\delta}$ a value associated with the right session where δ is the corresponding value in the left session.

1. *Let* $(\mathsf{View}, w_1, \ldots, w_{\mathsf{poly}(\lambda)})$ *denote the output of* $S(1^\lambda, z)$*, for some* $z \in \{0,1\}^\star$*. The probability ensembles* $\{S^1(1^\lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^\star}$ *and* $\{\mathsf{View}^{\mathcal{A}}(1^\lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^\star}$ *are computationally indistinguishable over* λ*, where* $S^1(1^\lambda, z)$ *denotes the first output of* $S(1^\lambda, z)$*.*
2. *For every* $i \in \{1, \ldots, \mathsf{poly}(\lambda)\}$*, if the* i*-th right session is accepting w.r.t. some statement* x_i *and* $\mathcal{A}$ *does not acts as a proxy (by simply sending back and forward the massages of the left session), then* w_i *is s.t.* $(x_i, w_i) \in \mathsf{Rel_L}$[6]*.*

The above definition of NMZK allows the adversary to select statements adaptively in the last round both in left and in the right sessions. Therefore any argument system that is NMZK according to the above definition enjoys also adaptive-input argument of knowledge.

3.3.2 Two-party Computation with a Simultaneous Message Exchange Channel

Our Two-Party Computation (2PC) protocol is secure in the same model used in [GMPP16, Pol16], therefore the following definition is taken almost verbatim from [GMPP16, Pol16].

A two-party protocol problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality and denote it $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ where $F = (F_1, F_2)$. That is, for every pair of inputs (x, y), the output-pair is a random variable $(F_1(x, y), F_2(x, y))$ ranging over pairs of strings. The first party (with input x) wishes to obtain $F_1(x, y)$ and the second party (with input y) wishes to obtain $F_2(x, y)$.

Adversarial behaviour. Loosely speaking, the aim of a secure two-party protocol is to protect an honest party against dishonest behaviour by the other party. In this work, we consider malicious adversaries who may arbitrarily deviate from the specified protocol. When considering malicious adversaries, there are certain undesirable actions that cannot be prevented. Specifically, a party may refuse to participate in the protocol, may substitute its local input (and use instead a different input) and may abort the protocol prematurely. One ramification of the adversary's ability to abort, is that it is impossible to achieve fairness. That is, the adversary may obtain its output while the honest party does not. In this work we consider a static corruption model, where one of the parties is adversarial and the other is honest, and this is fixed before the execution begins.

Communication channel. In our result we consider a secure simultaneous message exchange channel in which all parties can simultaneously send messages over the channel at the same communication round but allowing a rushing adversary. Moreover, we assume an asynchronous network[7] where the communication is open and delivery of messages is not guaranteed. For simplicity, we assume that the delivered messages are authenticated. This can be achieved using standard methods.

Execution in the ideal model. An ideal execution proceeds as follows. Each party obtains an input, denoted w ($w = x$ for P_1, and $w = y$ for P_2). An honest party always sends w to the trusted party. A malicious party may, depending on w, either abort or send some $w' \in \{0,1\}^{|w|}$ to the trusted party. In case it has obtained an input pair (x, y), the trusted party first replies to the first party with $F_1(x, y)$. Otherwise (i.e., in case it receives only one valid input), the

[6]In this definition we do not consider identities, since we do not need them for our propose of constructing a 2PC protocol.

[7]The fact that the network is asynchronous means that the messages are not necessarily delivered in the order which they are sent.

trusted party replies to both parties with a special symbol $\perp$. In case the first party is malicious it may, depending on its input and the trusted party's answer, decide to stop the trusted party by sending it $\perp$ after receiving its output. In this case the trusted party sends $\perp$ to the second party. Otherwise (i.e., if not stopped), the trusted party sends $F_2(x, y)$ to the second party. Outputs: an honest party always outputs the message it has obtained from the trusted party. A malicious party may output an arbitrary (probabilistic polynomial-time computable) function of its initial input and the message obtained from the trusted party.

Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a functionality where $F = (F_1, F_2)$ and let $S = (S_1, S_2)$ be a pair of non-uniform probabilistic expected polynomial-time machines (representing parties in the ideal model). Such a pair is admissible if for at least one $i \in \{0,1\}$ we have that S_i is honest (i.e., follows the honest party instructions in the above-described ideal execution). Then, the joint execution of F under S in the ideal model (on input pair (x, y) and security parameter λ), denoted $\mathsf{IDEAL}_{F,S(z)}(1^\lambda, x, y)$ is defined as the output pair of S_1 and S_2 from the above ideal execution.

Execution in the real model. We next consider the real model in which a real (two-party) protocol is executed (and there exists no trusted third party). In this case, a malicious party may follow an arbitrary feasible strategy; that is, any strategy implementable by non-uniform probabilistic polynomial-time machines. In particular, the malicious party may abort the execution at any point in time (and when this happens prematurely, the other party is left with no output). Let F be as above and let Π be a two-party protocol for computing F. Furthermore, let $A = (A_1, A_2)$ be a pair of non-uniform probabilistic polynomial-time machines (representing parties in the real model). Such a pair is admissible if for at least one $i \in \{0,1\}$ we have that A_i is honest (i.e., follows the strategy specified by Π). Then, the joint execution of Π under A in the real model, denoted $\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda)$, is defined as the output pair of A_1 and A_2 resulting from the protocol interaction.

Definition 10 (secure two-party computation)**.** *Let F and Π be as above. Protocol Π is said to securely compute F (in the malicious model) if for every pair of admissible non-uniform probabilistic polynomial-time machines $A = (A_1, A_2)$ that run with auxiliary input z for the real model, there exists a pair of admissible non-uniform probabilistic expected polynomial-time machines $S = (S_1, S_2)$ (that use z as auxiliary input) for the ideal model, such that:*

$$\{\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda, x, y)\}_{\lambda\in\mathbb{N},z,x,y\in\{0,1\}^\star} \approx \{\mathsf{IDEAL}_{f,S(z)}(1^\lambda, x, y)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^\star}.$$

We note that the above definition assumes that the parties know the input lengths (this can be seen from the requirement that $|x| = |y|$). Some restriction on the input lengths is unavoidable, see Section 7.1 of [Gol04] for discussion. We also note that we allow the ideal adversary/simulator to run in expected (rather than strict) polynomial-time. This is essential for constant-round protocols.

3.3.3 Oblivious Transfer

Here we follow [ORS15]. Oblivious Transfer (OT) is a two-party functionality $F_{\mathcal{OT}}$, in which a sender S holds a pair of strings (l_0, l_1), and a receiver R holds a bit b, and wants to obtain the string l_b. The security requirement for the $F_{\mathcal{OT}}$ functionality is that any malicious receiver does not learn anything about the string l_{1-b} and any malicious sender does not learn which string has been transferred. This security requirement is formalized via the ideal/real world paradigm. In the ideal world, the functionality is implemented by a trusted party that takes the inputs from S and R and provides the output to R and is therefore secure by definition. A real

Functionality $F_{\mathcal{OT}}$

$F_{\mathcal{OT}}$ running with a sender S a receiver R and an adversary Sim proceeds as follows:

- Upon receiving a message $(\mathsf{send}, l_0, l_1, S, R)$ from S where each $l_0, l_1 \in \{0,1\}^\lambda$, record the tuple (l_0, l_1) and send send to R and Sim. Ignore any subsequent send messages.
- Upon receiving a message $(\mathsf{receive}, b)$ from R, where $b \in \{0,1\}$ send l_b to R and $\mathsf{receive}$ to S and Sim and halt. (If no $(\mathsf{send}, \cdot)$ message was previously sent, do nothing).

Figure 3.4: The Oblivious Transfer Functionality $F_{\mathcal{OT}}$.

world protocol Π securely realizes the ideal $F_{\mathcal{OT}}$ functionalities, if the following two conditions hold. (a) Security against a malicious receiver: the output of any malicious receiver $R^\star$ running one execution of Π with an honest sender S can be simulated by a PPT simulator Sim that has only access to the ideal world functionality $F_{\mathcal{OT}}$ and oracle access to $R^\star$. (b) Security against a malicious sender. The joint view of the output of any malicious sender $S^\star$ running one execution of Π with R and the output of R can be simulated by a PPT simulator Sim that has only access to the ideal world functionality functionality $F_{\mathcal{OT}}$ and oracle access to $S^\star$. In this work we consider a weaker definition of $F_{\mathcal{OT}}$ that is called one-sided simulatable $F_{\mathcal{OT}}$, in which we do not demand the existence of a simulator against a malicious sender, but we only require that a malicious sender cannot distinguish whether the honest receiver is playing with bit 0 or 1. A bit more formally, we require that for any PPT malicious sender $S^\star$ the view of $S^\star$ executing Π with the R playing with bit 0 is computationally indistinguishable from the view of $S^\star$ where R is playing with bit 1. Finally, we consider the $F^m_{\mathcal{OT}}$ functionality where the sender S and the receiver R run m executions of OT in parallel. The formal definitions of one-sided secure $F_{\mathcal{OT}}$ and one-sided secure $F^m_{\mathcal{OT}}$ follow.

Definition 11 ([ORS15]). *Let $F_{\mathcal{OT}}$ be the Oblivious Transfer functionality as shown in Fig. 3.4. We say that a protocol Π securely computes $F_{\mathcal{OT}}$ with one-sided simulation if the following holds:*

1. *For every non-uniform* PPT *adversary $R^\star$ controlling the receiver in the real model, there exists a non-uniform* PPT *adversary Sim for the ideal model such that*

$$\{\mathsf{REAL}_{\Pi,R^\star(z)}(1^\lambda)\}_{z\in\{0,1\}^\lambda} \approx \mathsf{IDEAL}_{F_{\mathcal{OT}},\mathsf{Sim}(z)}(1^\lambda)\}_{z\in\{0,1\}^\lambda}$$

   where $\mathsf{REAL}_{\Pi,R^\star(z)}(1^\lambda)$ denotes the distribution of the output of the adversary $R^\star$ (controlling the receiver) after a real execution of protocol Π, where the sender S has inputs l_0, l_1 and the receiver has input b. $\mathsf{IDEAL}_{f,\mathsf{Sim}(z)}(1^\lambda)$ denotes the analogous distribution in an ideal execution with a trusted party that computes $F_{\mathcal{OT}}$ for the parties and hands the output to the receiver.

2. *For every non-uniform* PPT *adversary $S^\star$ controlling the sender it holds that:*

$$\{\mathsf{View}^R_{\Pi,S^\star(z)}(l_0, l_1, 0)\}_{z\in\{0,1\}^\star} \approx \{\mathsf{View}^R_{\Pi,S^\star(z)}(l_0, l_1, 1)\}_{z\in\{0,1\}^\star}$$

   where $\mathsf{View}^R_{\Pi,S^\star(z)}$ denotes the view of adversary $S^\star$ after a real execution of protocol Π with the honest receiver R.

Definition 12 (Parallel oblivious transfer functionality $F^m_{\mathcal{OT}}$ [ORS15]). *The parallel Oblivious Transfer Functionality $F^m_{\mathcal{OT}}$ is identical to the functionality $F_{\mathcal{OT}}$, with the difference that takes* *in input m pairs of string from S $(l_0^1, l_1^1, \dots, l_0^m, l_1^m)$ (whereas $F_{\mathcal{OT}}$ takes just one pair of strings from S) and m bits from R, $b_1, \dots, b_m$ (whereas $F_{\mathcal{OT}}$ takes one bit from R) and outputs to the receiver values $(l_{b_1}^1, \dots, l_{b_m}^m)$ while the sender receives nothing.*

Definition 13 ([ORS15]). *Let $F_{\mathcal{OT}}^m$ be the Oblivious Transfer functionality as described in Def. 12. We say that a protocol Π securely computes $F_{\mathcal{OT}}^m$ with one-sided simulation if the following holds:*

1. *For every non-uniform PPT adversary $R^\star$ controlling the receiver in the real model, there exists a non-uniform PPT adversary Sim for the ideal model such that for every $x_1 \in \{0,1\}, \dots, x_m \in \{0,1\}$*

$$\{\mathsf{REAL}_{\Pi,R^\star(z)}(1^\lambda, (l_0^1, l_1^1, \dots, l_0^m, l_1^m), (x_1, \dots, x_m))\} \approx \\ \mathsf{IDEAL}_{F_{\mathcal{OT}}^m,\mathsf{Sim}(z)}(1^\lambda), (l_0^1, l_1^1, \dots, l_0^m, l_1^m), (x_1, \dots, x_m))\}_{z \in \{0,1\}^\lambda}$$

   where $\mathsf{REAL}_{\Pi,R^\star(z)}(1^\lambda)$ denotes the distribution of the output of the adversary $R^\star$ (controlling the receiver) after a real execution of protocol Π, where the sender S has inputs $(l_0^1, l_1^1, \dots, l_0^m, l_1^m)$ and the receiver has input $(x_1, \dots, x_m)$. $\mathsf{IDEAL}_{f,\mathsf{Sim}(z)}(1^\lambda)$ denotes the analogous distribution in an ideal execution with a trusted party that computes $F_{\mathcal{OT}}^m$ for the parties and hands the output to the receiver.

2. *For every non-uniform PPT adversary $S^\star$ controlling the sender it holds that for every $x_1 \in \{0,1\}, \dots, x_m \in \{0,1\}$ and for every $y_1 \in \{0,1\}, \dots, y_m \in \{0,1\}$:*

$$\{\mathsf{View}_{\Pi,S^\star(z)}^R((l_0^1, l_1^1, \dots, l_0^m, l_1^m), (x_1, \dots, x_m))\}_{z \in \{0,1\}^\star} \approx \\ \{\mathsf{View}_{\Pi,S^\star(z)}^R((l_0^1, l_1^1, \dots, l_0^m, l_1^m), (y_1, \dots, y_m))\}_{z \in \{0,1\}^\star}$$

   where $\mathsf{View}_{\Pi,S^\star(z)}^R$ denotes the view of adversary $S^\star$ after a real execution of protocol Π with the honest receiver R.

We remark that in this notions of OT we do not suppose the existence of a simultaneous message exchange channel.

3.4 Our OT Protocol $\Pi_{\mathcal{OT}}^\gamma = (S_{\mathcal{OT}}, R_{\mathcal{OT}})$

We use the following tools.

1. A non-interactive perfectly binding, computationally hiding commitment scheme $\mathsf{PBCOM} = (\mathsf{Com}, \mathsf{Dec})$.
2. A trapdoor permutation $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$[8] with the hardcore bit function for λ bits $\mathsf{hc}(\cdot)$ (see Def. 8).
3. A non-interactive IDTC scheme $\mathsf{TC}_0 = (\mathsf{Sen}_0, \mathsf{Rec}_0, \mathsf{TFake}_0)$ for the $\mathcal{NP}$-language $L_0 = \{\mathtt{com} : \exists\ \mathtt{dec}$ s.t. $\mathsf{Dec}(\mathtt{com}, \mathtt{dec}, 0) = 1\}$.
4. A non-interactive IDTC scheme $\mathsf{TC}_1 = (\mathsf{Sen}_1, \mathsf{Rec}_1, \mathsf{TFake}_1)$ for the $\mathcal{NP}$-language $L_1 = \{\mathtt{com} : \exists\ \mathtt{dec}$ s.t. $\mathsf{Dec}(\mathtt{com}, \mathtt{dec}, 1) = 1\}$.

Let $b \in \{0,1\}$ be the input of $R_{\mathcal{OT}}$ and $l_0, l_1 \in \{0,1\}\lambda$ be the input of $S_{\mathcal{OT}}$, we now give the description of our protocol following Fig. 3.5.

[8]We recall that for convenience, we drop $(f, \mathtt{td})$ from the notation, and write $f(\cdot)$, $f^{-1}(\cdot)$ to denote algorithms $\mathsf{Eval}(f, \cdot)$, $\mathsf{Invert}(f, \mathtt{td}, \cdot)$ respectively, when f, $\mathtt{td}$ are clear from the context. Also we omit the generalization to a family of TDPs.

In the **first round** $R_{\mathcal{OT}}$ runs Com on input the message to be committed b in order to obtain the pair $(\mathtt{com}, \mathtt{dec})$. On input the instance $\mathtt{com}$ and a random string r^1_{b-1}, $R_{\mathcal{OT}}$ runs Sen_{1-b} in order to compute the pair $(\mathtt{tcom}_{1-b}, \mathtt{tdec}_{1-b})$. We observe that the *Instance-Dependent Binding* property of the IDTCs, the description of the $\mathcal{NP}$-language L_{1-b} and the fact that in $\mathtt{com}$ the bit b has been committed, ensure that $\mathtt{tcom}_{1-b}$ can be opened only to the value r^1_{b-1}.[9] $R_{\mathcal{OT}}$ runs the trapdoor procedure of the IDTC scheme TC_b. More precisely $R_{\mathcal{OT}}$ runs TFake_b on input the instance $\mathtt{com}$ to compute the pair $(\mathtt{tcom}_b, \mathtt{aux})$. In this case $\mathtt{tcom}_b$ can be equivocated to any message using the trapdoor (the opening information of $\mathtt{com}$), due to the trapdoorness of the IDTC, the description of the $\mathcal{NP}$-language L_b and the message committed in $\mathtt{com}$ (that is represented by the bit b). $R_{\mathcal{OT}}$ sends $\mathtt{tcom}_0$, $\mathtt{tcom}_1$ and $\mathtt{com}$ to $S_{\mathcal{OT}}$.

In the **second round** $S_{\mathcal{OT}}$ picks two random strings R_0, R_1 and two trapdoor permutations $(f_{0,1}, f_{1,1})$ along with their trapdoors $(f^{-1}_{0,1}, f^{-1}_{1,1})$. Then $S_{\mathcal{OT}}$ sends R_0, R_1, $f_{0,1}$ and $f_{1,1}$ to $R_{\mathcal{OT}}$.

In the **third round** $R_{\mathcal{OT}}$ checks whether or not $f_{0,1}$ and $f_{1,1}$ are valid trapdoor permutations. In the negative case $R_{\mathcal{OT}}$ aborts, otherwise $R_{\mathcal{OT}}$ continues with the following steps. $R_{\mathcal{OT}}$ picks a random string z'_1 and computes $z_1 = f_{b,1}(z'_1)$. $R_{\mathcal{OT}}$ now computes $r^1_b = z_1 \oplus R_b$ and runs TFake_b on input $\mathtt{dec}$, $\mathtt{com}$, $\mathtt{tcom}_b$, $\mathtt{aux}$ and r^1_b in order to obtain the equivocal opening $\mathtt{tdec}_b$ of the commitment $\mathtt{tcom}_b$ to the message r^1_b. $R_{\mathcal{OT}}$ renames r_b to r^1_b and $\mathtt{tdec}_b$ to $\mathtt{tdec}^1_b$ and sends to $S_{\mathcal{OT}}$ $(\mathtt{tdec}^1_0, r^1_0)$ and $(\mathtt{tdec}^1_1, r^1_1)$.

In the **fourth round** $S_{\mathcal{OT}}$ checks whether or not $(\mathtt{tdec}^1_0, r^1_0)$ and $(\mathtt{tdec}^1_1, r^1_1)$ are valid openings w.r.t. $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$. In the negative case $S_{\mathcal{OT}}$ aborts, otherwise $S_{\mathcal{OT}}$ computes $W^1_0 = l_0 \oplus \mathsf{hc}(f^{-\lambda}_{0,1}(r^1_0 \oplus R_0))$ and $W^1_1 = l_1 \oplus \mathsf{hc}(f^{-\lambda}_{1,1}(r^1_1 \oplus R_1))$. Informally $S_{\mathcal{OT}}$ encrypts his inputs l_0 and l_1 through a one-time pad using as a secret key the pre-image of $r^1_0 \oplus R_0$ for l_0 and the pre-image of $r^1_1 \oplus R_1$ for l_1. $S_{\mathcal{OT}}$ also computes two trapdoor permutations $(f_{0,2}, f_{1,2})$ along with their trapdoors $(f^{-1}_{0,2}, f^{-1}_{1,2})$ and sends $(W^1_0, W^1_1, f_{0,2}, f_{1,2})$ to $R_{\mathcal{OT}}$. At this point the third and the fourth rounds are repeated up to $\gamma - 1$ times using fresh randomness as showed in Fig. 3.5. In the last round no trapdoor permutations are needed/sent.

We observe that a malicious sender $S^\star_{\mathcal{OT}}$ could easily understand the input bit of $R_{\mathcal{OT}}$ when $\gamma > 1$. This is not a problem since for our application we need to prove the security of $\Pi^\gamma_{\mathcal{OT}}$ to hold against malicious sender only for $\gamma = 1$. We only consider $\gamma = \mathsf{poly}(\lambda)$ when proving the security of $\Pi^\gamma_{\mathcal{OT}}$ against malicious receiver.

In the **output phase**, $R_{\mathcal{OT}}$ computes and outputs $l_b = W^1_b \oplus \mathsf{hc}(z'_1)$. That is, $R_{\mathcal{OT}}$ just uses the information gained in the fourth round to compute the output. It is important to observe that $R_{\mathcal{OT}}$ can correctly and efficiently compute the output because $z' = r^1_b \oplus R_b$. Moreover $R_{\mathcal{OT}}$ cannot compute l_{1-b} because he has no way to change the value committed in $\mathtt{tcom}_{1-b}$ and invert the TDP since it is suppose to be hard without having the trapdoor.

In order to construct our protocol for two-party computation in the simultaneous message exchange model we need to consider an extended version of $\Pi^\gamma_{\mathcal{OT}}$, that we denote by $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}} = (S_{\overrightarrow{\mathcal{OT}}}, R_{\overrightarrow{\mathcal{OT}}})$. In $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ the $S_{\overrightarrow{\mathcal{OT}}}$'s input is represented by m pairs $(l^1_0, l^1_1, \dots, l^m_0, l^m_1)$ and the $R_{\overrightarrow{\mathcal{OT}}}$'s input is represented by the sequence $b_1, \dots, b_m$ with $b_i \in \{0, 1\}$ for all $i = 1, \dots, m$. In this case the output of $R_{\overrightarrow{\mathcal{OT}}}$ is $(l_{b_1}, \dots, l_{b_m})$. We construct $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}} = (S_{\overrightarrow{\mathcal{OT}}}, R_{\overrightarrow{\mathcal{OT}}})$ by simply considering m parallel iterations of $\Pi^\gamma_{\mathcal{OT}}$ and then we prove that it securely computes $F^m_{\mathcal{OT}}$ with one-sided simulation (see Definition 13).

Proof sketch. The security proof of $\Pi^\gamma_{\mathcal{OT}}$ is divided in two parts. In the former we prove the security against a malicious sender with $\gamma = 1$ and in the latter we prove the security of $\Pi^\gamma_{\mathcal{OT}}$ against a malicious receiver with $\gamma = \mathsf{poly}(\lambda)$. In order to prove the security against malicious sender we recall that for the definition of one-sided simulation it is just needed the no

[9] $\mathtt{com}$ does not belong to the $\mathcal{NP}$-language L_{b-1}, therefore $\mathtt{tcom}_{1-b}$ is a perfectly binding commitment.

$R_{\mathcal{OT}}(b)$ $\qquad\qquad$ $S_{\mathcal{OT}}(l_0, l_1)$

$(\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{Com}(1^\lambda, b)$;
$(\mathtt{tcom}_b, \mathtt{aux}) \leftarrow \mathsf{TFake}_b(1^\lambda, \mathtt{com})$;
$r_{1-b} \leftarrow \{0,1\}^\lambda$;
$(\mathtt{tcom}_{1-b}, \mathtt{tdec}_{1-b}) \leftarrow \mathsf{Sen}_{1-b}(1^\lambda, r_{1-b}, \mathtt{com})$.

$$\xrightarrow{\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1}$$

$R_0 \leftarrow \{0,1\}^\lambda$;
$R_1 \leftarrow \{0,1\}^\lambda$;
$(f_{0,1}, f_{0,1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$;
$(f_{1,1}, f_{1,1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$.

$$\xleftarrow{R_0, R_1, f_{0,1}, f_{1,1}}$$

$z'_1 \leftarrow \{0,1\}^\lambda$;
$z_1 = f_{b,1}^\lambda(z'_1)$;
$r_b^1 = z_1 \oplus R_b$;
$\mathtt{tdec}_b^1 \leftarrow \mathsf{TFake}_b(\mathtt{dec}, \mathtt{com}, \mathtt{tcom}_b, \mathtt{aux}, r_b^1)$;
$\mathtt{tdec}_{1-b}^1 = \mathtt{tdec}_{1-b}$, $r_{1-b}^1 = r_{1-b}$.

$$\xrightarrow{(\mathtt{tdec}_0^1, r_0^1), (\mathtt{tdec}_1^1, r_1^1)}$$

$(f_{0,2}, f_{0,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$;
$(f_{1,2}, f_{1,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$;
$W_0^1 = l_0 \oplus \mathsf{hc}(f_{0,1}^{-\lambda}(r_0^1 \oplus R_0))$;
$W_1^1 = l_1 \oplus \mathsf{hc}(f_{1,1}^{-\lambda}(r_1^1 \oplus R_1))$.

$$\xleftarrow{W_0^1, W_1^1, f_{0,2}, f_{1,2}}$$

$z'_2 \leftarrow \{0,1\}^\lambda$;
$z_2 = f_{b,2}^\lambda(z'_2)$;
$r_b^2 = z_2 \oplus R_b$;
$\mathtt{tdec}_b^2 \leftarrow \mathsf{TFake}_b(\mathtt{dec}, \mathtt{com}, \mathtt{tcom}_b, \mathtt{aux}, r_b^2)$;
$\mathtt{tdec}_{1-b}^2 = \mathtt{tdec}_{1-b}$, $r_{1-b}^2 = r_{1-b}$.

$$\xrightarrow{(\mathtt{tdec}_0^2, r_0^2), (\mathtt{tdec}_1^2, r_1^2)}$$

$(f_{0,3}, f_{0,3}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$;
$(f_{1,3}, f_{1,3}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$;
$W_0^2 = l_0 \oplus \mathsf{hc}(f_{0,2}^{-\lambda}(r_0^2 \oplus R_0))$;
$W_1^2 = l_1 \oplus \mathsf{hc}(f_{1,2}^{-\lambda}(r_1^2 \oplus R_1))$.

$$\xleftarrow{W_0^2, W_1^2, f_{0,3}, f_{1,3}}$$

$$\vdots$$

$$\xrightarrow{(\mathtt{tdec}_0^\gamma, r_0^\gamma), (\mathtt{tdec}_1^\gamma, r_1^\gamma)}$$

$$\xleftarrow{W_0^\gamma, W_1^\gamma}$$

Output $l_b = W_b^1 \oplus \mathsf{hc}(z'_1)$.

Figure 3.5: Description of $\Pi_{\mathcal{OT}}^\gamma$.

information about R's input is leaked to $S^\star$. We consider the experiment H_0 where R's input is 0 and the experiment H_1 where R's input is 1 and we prove that $S^\star$ cannot distinguish between H_0 and H_1. More precisely we consider the experiment H^a where $\mathtt{tcom}_0$ and the corresponding opening is computed without using the trapdoor (the randomness of $\mathtt{com}$) and relying on the trapdoorness of the IDTCom TC_0 we prove that $H_0 \approx H^a$. Then we consider the experiment H^b where the value committed in $\mathtt{com}$ goes from 0 to 1 and prove that $H^a \approx H^b$ due to the hiding of $\mathtt{com}$. We observe that this reduction can be made because to compute both H^a and H^b the opening informations of $\mathtt{com}$ are not required anymore. The proof ends with the observation that $H^b \approx H_1$ due to the trapdoorness of the IDTCom TC_1.

To prove the security against a malicious receiver $R^\star$ we need to show a simulator Sim. Sim rewinds $R^\star$ from the third to the second round by sending every time freshly generated R_0 and R_1. Sim then checks whether the values r_0^1 and r_1^1 change during the rewinds. We recall that $\mathtt{com}$ is a perfectly binging commitment, therefore only one between $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ can be opened to multiple values using the trapdoor procedure ($\mathtt{com}$ can belong only to one of the $\mathcal{NP}$-languages L_0 and L_1). Moreover, intuitively, the only way that $R^\star$ can compute the output is by equivocating one between $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ based on the values R_0, R_1 received in the second round. This means that if during the rewinds the value opened w.r.t. $\mathtt{tcom}_b$ changes, then the input that $R^\star$ is using is b. Therefore the simulator can call the ideal functionality thus obtaining l_b. At this point Sim uses l_b to compute W_b^1 according to the description of $\Pi^\gamma_{\mathcal{OT}}$ and sets W_{1-b}^1 to a random string. Moreover Sim will use the same strategy used to compute W_b^1 and W_{1-b}^1 to compute, respectively W_b^i and W_{1-b}^i for $i = 2, \dots, \gamma$. In case during the rewinds the value r_0^1, r_1^1 stay the same, then Sim sets both W_0^1 and W_1^1 to random strings. We observe that $R^\star$ could detect that now W_0^1 and W_1^1 are computed in a different way, but this would violate the security of the TDPs.

Theorem 1. *Assuming TDPs, for any $\gamma > 0$ $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ securely computes $F^m_{\mathcal{OT}}$ with one-sided simulation. Moreover the third round is* replayable.

Proof. We first observe that in third round of $\Pi^\gamma_{\mathcal{OT}}$ only the opening information for the IDTCs $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ are sent. Therefore once that a valid third round is received, it is possible to replay it in order to answer to many second rounds sent by a malicious sender. Roughly, whether the third round of $\Pi^\gamma_{\mathcal{OT}}$ is accepting or not is independent of what a malicious sender sends in the second round. Therefore we have proved that $\Pi^\gamma_{\mathcal{OT}}$ has a *replayable* third round. In order to prove that $\Pi^\gamma_{\mathcal{OT}}$ is one-sided simulatable secure for $F_{\mathcal{OT}}$ (see Definition 11) we divide the security proof in two parts; the former proves the security against a malicious sender, and the latter proves the security against a malicious receiver. More precisely we prove that $\Pi^\gamma_{\mathcal{OT}}$ is secure against a malicious receiver for an arbitrary chosen $\gamma = \mathsf{poly}(\lambda)$, and is secure against malicious sender for $\gamma = 1$ (i.e. when just the first four rounds of the protocol are executed).

Security against a malicious sender. In this case we just need to prove that the output of $S^\star_{\mathcal{OT}}$ of the execution of $\Pi^\gamma_{\mathcal{OT}}$ when $R_{\mathcal{OT}}$ interacts with $S^\star_{\mathcal{OT}}$ using $b = 0$ as input is computationally indistinguishable from when $R_{\mathcal{OT}}$ uses $b = 1$ as input. The differences between these two hybrid experiments consist of the message committed in $\mathtt{com}$ and the way in which the IDTCs are computed. More precisely, in the first experiment, when $b = 0$ is used as input, $\mathtt{tcom}_0$ and the corresponding opening $(\mathtt{tdec}_0^1, r_0^1)$ are computed using the trapdoor procedure (in this case the message committed in $\mathtt{com}$ is 0), while $\mathtt{tcom}_1$ and $(\mathtt{tdec}_1^1, r_1^1)$ are computed using the *honest* procedure. In the second experiment, $\mathtt{tcom}_0$ and the respective opening $(\mathtt{tdec}_0^1, r_0^1)$ are computed using the honest procedure, while $\mathtt{tcom}_1$ and $(\mathtt{tdec}_1^1, r_1^1)$ are computed using the trapdoor procedure of the IDTC scheme. In order to prove the indistinguishability between these two experiments we proceed via hybrid arguments. The first hybrid experiment $\mathcal{H}_1$ is equal to when $R_{\mathcal{OT}}$ interacts with against $S^{\star}_{\mathcal{OT}}$ according $\Pi^{\gamma}_{\mathcal{OT}}$ when $b = 0$ is used as input. In $\mathcal{H}_2$ the honest procedure of IDTC is used instead of the trapdoor one in order to compute $\mathtt{tcom}_0$ and the opening $(\mathtt{tdec}^1_0, r^1_0)$. We observe that in $\mathcal{H}_2$ both the IDTCs are computed using the honest procedure, therefore no trapdoor information (i.e. the randomness used to compute $\mathtt{com}$) is required. The computational-indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ comes from the trapdoorness of the IDTC TC_0. In $\mathcal{H}_3$ the value committed in $\mathtt{com}$ goes from 0 to 1. $\mathcal{H}_2$ and $\mathcal{H}_3$ are indistinguishable due to the hiding of PBCOM. It is important to observe that a reduction to the hiding of PBCOM is possible because the randomness used to compute $\mathtt{com}$ is no longer used in the protocol execution to run one of the IDTCs. In the last hybrid experiment $\mathcal{H}_4$ the trapdoor procedure is used in order to compute $\mathtt{tcom}_1$ and the opening $(\mathtt{tdec}^1_1, r^1_1)$. We observe that it is possible to run the trapdoor procedure for TC_1 because the message committed in $\mathtt{com}$ is 1. The indistinguishability between $\mathcal{H}_3$ and $\mathcal{H}_4$ comes from the trapdoorness of the IDTC. The observation that $\mathcal{H}_4$ corresponds to the experiment where the honest receiver executes $\Pi^{\gamma}_{\mathcal{OT}}$ using $b = 1$ as input concludes the security proof.

Security against a malicious receiver. In order to prove that $\Pi^{\gamma}_{\mathcal{OT}}$ is simulation-based secure against malicious receiver $R^{\star}_{\mathcal{OT}}$ we need to show a PPT simulator Sim that, having only access to the ideal world functionality $F_{\mathcal{OT}}$, can simulate the output of any malicious $R^{\star}_{\mathcal{OT}}$ running one execution of $\Pi^{\gamma}_{\mathcal{OT}}$ with an honest sender $S_{\mathcal{OT}}$. The simulator Sim works as follows. Having oracle access to $R^{\star}_{\mathcal{OT}}$, Sim runs as a sender in $\Pi^{\gamma}_{\mathcal{OT}}$ by sending two random strings R_0 and R_1 and the pair of TDPs $f_{0,1}$ and $f_{1,1}$ in the second round. Let $(\mathtt{tdec}^1_0, r^1_0), (\mathtt{tdec}^1_1, r^1_1)$ be the messages sent in the third round by $R^{\star}_{\mathcal{OT}}$. Now Sim rewinds $R^{\star}_{\mathcal{OT}}$ by sending two fresh random strings $\overline{R}_0$ and $\overline{R}_1$ such that $\overline{R}_0 \neq R_0$ and $\overline{R}_1 \neq R_1$.

Let $(\overline{\mathtt{tdec}}^1_0, \overline{r}^1_0), (\overline{\mathtt{tdec}}^1_1, \overline{r}^1_1)$ be the messages sent in the third round by $R^{\star}_{\mathcal{OT}}$ after this rewind, then there are only two things that can happen[10]:

1. $r^1_{b^\star} \neq \overline{r}^1_{b^\star}$ and $r^1_{1-b^\star} = \overline{r}^1_{1-b^\star}$ for some $b^\star \in \{0,1\}$ or
2. $r^1_0 = \overline{r}^1_0$ and $r^1_1 = \overline{r}^1_1$.

More precisely, due to the perfect binding of PBCOM at most one between $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ can be opened to a different message. Therefore $R^{\star}_{\mathcal{OT}}$ can either open both $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ to the same messages r^1_0 and r^1_1, or change in the opening of at most one of them. This yields to the following important observation. If one among r^1_0 and r^1_1 changes during the rewind, let us say $r_{b^\star}$ for $b^\star \in \{0,1\}$ (case 1), then the input bit used by $R^{\star}_{\mathcal{OT}}$ has to be $b^\star$. Indeed we recall that the only efficient way (i.e. without inverting the TDP) for a receiver to get the output is to equivocate one of the IDTCs in order to compute the inverse of one between $R_0 \oplus r^1_0$ and $R_1 \oplus r^1_1$. Therefore the simulator invokes the ideal world functionality $F_{\mathcal{OT}}$ using $b^\star$ as input, and upon receiving $l_{b^\star}$ computes $W^1_{b^\star} = l_{b^\star} \oplus \mathsf{hc}(f^{-\lambda}_{b^\star,1}(r^1_{b^\star} \oplus R_{b^\star}))$ and sets $W^1_{1-b^\star}$ to a random string. Then sends W^1_0 and W^1_1 with two freshly generated TDPs $f_{0,2}, f_{1,2}$ (according to the description of $\Pi^{\gamma}_{\mathcal{OT}}$ given in Fig. 3.5) to $R^{\star}_{\mathcal{OT}}$. Let us now consider the case where the opening of $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ stay the same after the rewinding procedure (case two). In this case, Sim comes back to the main thread and sets both W^1_0 and W^1_1 to a random string. Intuitively if $R^{\star}_{\mathcal{OT}}$ does not change neither r^1_0 nor r^1_1 after the rewind, then his behavior is not adaptive on the second round sent by Sim. Therefore, he will be able to compute the inverse of neither $R_0 \oplus r^1_0$ nor $R_1 \oplus r^1_1$. That is, both $R_0 \oplus r^1_0$ and $R_1 \oplus r^1_1$ would be the results of the execution of two coin-flipping protocols, therefore both of them are difficult to invert without knowing the trapdoors of the TDPs. This implies that $R^{\star}_{\mathcal{OT}}$ has no efficient way to tells apart whether W^1_0 and W^1_1 are random strings or not.

Completed the fourth round, for $i = 2, \ldots, \gamma$, Sim continues the interaction with $R^{\star}_{\mathcal{OT}}$ by

[10] $R^{\star}_{\mathcal{OT}}$ could also abort after the rewind. In this case we use the following standard argument. If p is the probability of $R^{\star}_{\mathcal{OT}}$ of giving an accepting third round, λ/p rewinds are made until $R^{\star}_{\mathcal{OT}}$ gives another answer.

always setting both W_0^i and W_1^i to a random string when $r_0^1 = r_0^i$ and $r_1^1 = r_1^i$, and using the following strategy when $r_{b^\star}^1 \neq r_{b^\star}^i$ and $r_{1-b^\star}^1 = r_{1-b^\star}^i$ for some $b^\star \in \{0,1\}$. Sim invokes the ideal world functionality $F_{\mathcal{OT}}$ using $b^\star$ as input, and upon receiving $l_{b^\star}$ computes $W_{b^\star}^i = l_{b^\star} \oplus \mathsf{hc}(f_{b^\star,i}^{-\lambda}(r_{b^\star}^i \oplus R_{b^\star}))$, sets $W_{1-b^\star}^i$ to a random string and sends with them two freshly generated TDPs $f_{0,i+1}, f_{1,i+1}$ to $R_{\mathcal{OT}}^\star$. When the interaction against $R_{\mathcal{OT}}^\star$ is over, Sim stops and outputs what $R_{\mathcal{OT}}^\star$ outputs. We observe that the simulator needs to invoke the ideal world functionality just once. Indeed, we recall that only one of the IDTCs can be equivocated, therefore once that the bit $b^\star$ is decided (using the strategy described before) it cannot change during the simulation. The last thing that remains to observe is that it could happen that Sim never needs to invoke the ideal world functionality in the case that: 1) during the rewind the values (r_0^1, r_1^1) stay the same; 2) $r_b^i = r_b^j$ for all $i,j \in \{1,\dots,\gamma\}$ and all $b = \{0,1\}$. In this case Sim, even though it does not need to query the ideal functionality to internally complete an interaction with $R_{\mathcal{OT}}^\star$ we assume, without loss of generality, that Sim invokes the ideal functionality by using a random bit $b^\star \in \{0,1\}$.

We formally prove that the output of Sim is computationally indistinguishable from the output of $R_{\mathcal{OT}}^\star$ in the real world execution for every $\gamma = \mathsf{poly}(\lambda)$. The proof goes trough hybrid arguments starting from the real world execution. We gradually modify the real world execution until the input of the honest party is not needed anymore such that the final hybrid would represent the simulator for the ideal world. We denote by $\mathsf{OUT}_{\mathcal{H}_i, R_{\mathcal{OT}}^\star(z)}(1^\lambda)$ the output distribution of $R_{\mathcal{OT}}^\star$ in the hybrid experiment $\mathcal{H}_i$.

- $\mathcal{H}_0$ is identical to the real execution. More precisely $\mathcal{H}_0$ runs $R_{\mathcal{OT}}^\star$ using fresh randomness and interacts with him as the honest sender would do on input (l_0, l_1).
- $\mathcal{H}_0^{\mathsf{rew}}$ proceeds according to $\mathcal{H}_0$ with the difference that $R_{\mathcal{OT}}^\star$ is rewound up to the second round by receiving two fresh random strings $\overline{R}_0$ and $\overline{R}_1$. This process is repeated until $R_{\mathcal{OT}}^\star$ completes the third round again (every time using different randomness). More precisely, if $R_{\mathcal{OT}}^\star$ aborts after the rewind then a fresh second round is sent up to λ/p times, where p is the probability of $R_{\mathcal{OT}}^\star$ of completing the third round in $\mathcal{H}_0$. If $p = \mathsf{poly}(\lambda)$ then the expected running time of $\mathcal{H}_0^{\mathsf{rew}}$ is $\mathsf{poly}(\lambda)$ and its output is statistically close to the output of $\mathcal{H}_0$. When the third round is completed the hybrid experiment comes back to the main thread and continues according to $\mathcal{H}_0$
- $\mathcal{H}_1$ proceeds according to $\mathcal{H}_0^{\mathsf{rew}}$ with the difference that after the rewinds executes the following steps. Let r_0^1 and r_1^1 be the messages opened by $R_{\mathcal{OT}}^\star$ in the third round of the main thread and $\overline{r}_0^1$ and $\overline{r}_1^1$ be the messages opened during the rewind. We distinguish two cases that could happen:
  1. $r_0^1 = \overline{r}_0^1$ and $r_1^1 = \overline{r}_1^1$ or
  2. $r_{b^\star}^1 \neq \overline{r}_{b^\star}^1$ and $\overline{r}_{1-b^\star}^1 = r_{1-b^\star}^1$ for some $b^\star \in \{0,1\}$.

  In this hybrid we assume that the first case happen with non-negligible probability. After the rewind $\mathcal{H}_1$ goes back to the main thread, and in order to compute the fourth round, picks $W_0^1 \leftarrow \{0,1\}^\lambda$ computes $W_1^1 = l_1 \oplus \mathsf{hc}(f_{1,1}^{-\lambda}(r_1^1 \oplus R_1))$, $(f_{0,2}, f_{0,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$, $(f_{1,2}, f_{1,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends $(W_0^1, W_1^1, f_{0,2}, f_{1,2})$ to $R_{\mathcal{OT}}^\star$. Then the experiment continues according to $\mathcal{H}_0$. Roughly, the difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ is that in the latter hybrid experiment W_0^1 is a random string whereas in $\mathcal{H}_1$ $W_0^1 = l_0 \oplus \mathsf{hc}(f_{0,1}^{-\lambda}(r_0^1 \oplus R_0))$.

  We now prove that the indistinguishability between $\mathcal{H}_0$ and $\mathcal{H}_1$ comes from the security of the hardcore bit function for λ bits hc for the TDP $\mathcal{F}$. More precisely, assuming by contradiction that the outputs of $\mathcal{H}_0$ and $\mathcal{H}_1$ are distinguishable we construct and adversary $\mathcal{A}^\mathcal{F}$ that distinguishes between the output of $\mathsf{hc}(x)$ and a random string of λ bits having as input $f^\lambda(x)$. Consider an execution where $R_{\mathcal{OT}}^\star$ has non-negligible advantage in distinguishing $\mathcal{H}_0^{\mathsf{rew}}$ from $\mathcal{H}_1$ and consider the randomness ρ used by $R_{\mathcal{OT}}^\star$ and the first round computed by $R_{\mathcal{OT}}^\star$ in this execution, let us say $\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1$. $\mathcal{A}^\mathcal{F}$, on input the randomness ρ, the messages r_0^1 and r_1^1 executes the following steps.

1. Start $R^\star_{\mathcal{OT}}$ with randomness ρ.
2. Let $(f, H, f^\lambda(x))$ be the challenge. Upon receiving the first round $(\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1)$ by $R^\star_{\mathcal{OT}}$, compute $R_0 = r_0^1 \oplus f^\lambda(x)$, pick a random string R_1, compute $(f_{1,1}, f_{1,1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$, set $f_{0,1} = f$ and sends $R_0, R_1, f_{0,1}, f_{1,1}$ to $R^\star_{\mathcal{OT}}$.
3. Upon receiving $(\mathtt{tdec}_0^1, r_0^1), (\mathtt{tdec}_1^1, r_1^1)$ compute $W_0^1 = l_0 \oplus H$, $W_1^1 = l_1 \oplus \mathsf{hc}(f_{1,1}^{-\lambda}(r_1^1 \oplus R_1))$, $(f_{0,2}, f_{0,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$, $(f_{1,2}, f_{1,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and send $(W_0^1, W_1^1, f_{0,2}, f_{1,2})$. [11]
4. Continue the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}_1$ (and $\mathcal{H}_0^{\mathsf{rew}}$) and output what $R^\star_{\mathcal{OT}}$ outputs.

This part of the security proof ends with the observation that if $H = \mathsf{hc}(x)$ then $R^\star_{\mathcal{OT}}$ acts as in $\mathcal{H}_0^{\mathsf{rew}}$, otherwise $R^\star_{\mathcal{OT}}$ acts as in $\mathcal{H}_1$.

– $\mathcal{H}_2$ proceeds according to $\mathcal{H}_1$ with the difference that both W_0 and W_1 are set to random strings. Also in this case the indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2$ comes from the security of the hardcore bit function for λ bits hc for the family $\mathcal{F}$ (the same arguments of the previous security proof can be used to prove the indistinguishability between $\mathcal{H}_2$ and $\mathcal{H}_1$).

– $\mathcal{H}_3$ In this hybrid experiment we consider the case where after the rewind, with non-negligible probability, $r_{b^\star}^1 \neq \overline{r}_{b^\star}^1$ and $\overline{r}_{1-b^\star}^1 = r_{1-b^\star}^1$ for some $b^\star \in \{0,1\}$.

In this case, in the main thread the hybrid experiment computes $W_{b^\star}^1 = l_{b^\star} \oplus \mathsf{hc}(f_{b^\star,1}^{-\lambda}(r_{b^\star}^1 \oplus R_{b^\star}))$, picks $W_{1-b^\star}^1 \leftarrow \{0,1\}^\star$ sends W_0^1, W_1^1 with two freshly generated TDPs $f_{0,2}, f_{1,2}$. $\mathcal{H}_3$ now continues the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}_2$. The indistinguishability between $\mathcal{H}_2$ and $\mathcal{H}_3$ comes from the security of the hardcore bit function for λ bits hc for the TDP $\mathcal{F}$. More precisely, assuming by contradiction that $\mathcal{H}_2$ and $\mathcal{H}_3$ are distinguishable, we construct and adversary $\mathcal{A}^\mathcal{F}$ that distinguishes between the output of $\mathsf{hc}(x)$ and a random string of λ bits having as input $f^\lambda(x)$. Consider an execution where $R^\star_{\mathcal{OT}}$ has non-negligible advantage in distinguish $\mathcal{H}_2$ from $\mathcal{H}_3$ and consider the randomness ρ used by $R^\star_{\mathcal{OT}}$ and the first round computed in this execution, let us say $\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1$. $\mathcal{A}^\mathcal{F}$, on input the randomness ρ, the message $b^\star$ committed in $\mathtt{com}$ and the message $r_{1-b^\star}^1$ committed $\mathtt{tcom}_{1-b^\star}$, $\mathcal{A}^\mathcal{F}$ executes the following steps.

1. Start $R^\star_{\mathcal{OT}}$ with randomness ρ.
2. Let $(f, H, f^\lambda(x))$ be the challenge. Upon receiving the first round $(\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1)$ by $R^\star_{\mathcal{OT}}$, compute $R_{1-b^\star} = r_{1-b^\star}^1 \oplus f^\lambda(x)$, pick a random string $R_{b^\star}$, computes $(f_{b^\star,1}, f_{b^\star,1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$, sets $f_{1-b^\star,1} = f$ and send $(R_0, R_1, f_{0,1}, f_{1,1})$ to $R^\star_{\mathcal{OT}}$.
3. Upon receiving $(\mathtt{tdec}_0^1, r_0^1), (\mathtt{tdec}_1^1, r_1^1)$ compute $W_{1-b^\star}^1 = l_{1-b^\star} \oplus H$, $W_{b^\star}^1 = l_{b^\star} \oplus \mathsf{hc}(f_{b^\star,1}^{-\lambda}(r_{b^\star}^1 \oplus R_{b^\star}))$, $(f_{0,2}, f_{0,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$, $(f_{1,2}, f_{1,2}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and send $(W_0^1, W_1^1, f_{0,2}, f_{1,2})$.
4. Continue the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}_2$ (and $\mathcal{H}_3$) and output what $R^\star_{\mathcal{OT}}$ outputs.

This part of the security proof ends with the observation that if $H = \mathsf{hc}(x)$ then $R^\star_{\mathcal{OT}}$ acts as in $\mathcal{H}_2$, otherwise he acts as in $\mathcal{H}_3$.

– $\mathcal{H}_3^j$ proceeds according to $\mathcal{H}_3$ with the differences that for $i = 2, \dots, j$
  1. if $r_{b^\star}^i \neq r_{b^\star}^1$ for some $b^\star \in \{0,1\}$ then $\mathcal{H}_3^j$ picks $W_{1-b^\star}^i \leftarrow \{0,1\}^\lambda$, computes $W_{b^\star}^i = l_{b^\star} \oplus \mathsf{hc}(f_{b^\star,i}^{-\lambda}(r_{b^\star}^i \oplus R_{b^\star}))$ and sends W_0^i, W_i^i with two freshly generated TDPs $f_{0,i+1}, f_{1,i+1}$ to $R^\star_{\mathcal{OT}}$ otherwise

[11]Observe that $R^\star_{\mathcal{OT}}$ could send values different from r_0^1 and r_1^1 in the third round. In this case $\mathcal{A}^\mathcal{F}$ just recomputes the second round using fresh randomness and asking another challenge $\overline{f}, \overline{H}, \overline{f}^\lambda(x)$ to the challenger until in the third round the messages r_0^1 and r_1^1 are received again. This allows $\mathcal{A}^\mathcal{F}$ to break the security of $\overline{f}$ because we are assuming that in this experiment $R^\star_{\mathcal{OT}}$ opens, with non-negligible probability, $\mathtt{tcom}_0$ to r_0^1 and $\mathtt{tcom}_1$ to r_1^1.

2. $\mathcal{H}_3^j$ picks $W_0^i \leftarrow \{0,1\}^\lambda$ and $W_1^i \leftarrow \{0,1\}^\lambda$ and sends W_0^i, W_1^i with two freshly generated TDPs $f_{0,i+1}, f_{1,i+1}$ to $R^\star_{\mathcal{OT}}$.

Roughly speaking, if $R^\star_{\mathcal{OT}}$ changes the opened message w.r.t. $\mathtt{tcom}_{b^\star}$, then $W_{b^\star}^i$ is correctly computed and $W_{1-b^\star}^i$ is sets to a random string. Otherwise, if the opening of $\mathtt{tcom}_0$ and $\mathtt{tcom}_1$ stay the same as in the third round, then both W_0^i and W_1^i are random strings (for $i = 2, \dots, j$). We show that $\mathsf{OUT}_{\mathcal{H}_3^{j-1}, R^\star_{\mathcal{OT}}(z)}(1^\lambda) \approx \mathsf{OUT}_{\mathcal{H}_3^{j}, R^\star_{\mathcal{OT}}(z)}(1^\lambda)$ in two steps. In the first step we show that the indistinguishability between these two hybrid experiments holds for the first case (when $r_{b^\star}^i \neq r_{b^\star}^1$ for some bit $b^\star$), and in the second step we show that the same holds when $r_0^i = r_0^1$ and $r_1^i = r_1^1$.

We first recall that if $r_{b^\star}^i \neq r_{b^\star}^1$, then $\mathtt{tcom}_{1-b^\star}$ is perfectly binding, therefore we have that $r_{1-b^\star}^i = r_{1-b^\star}^1$. Assuming by contradiction that $\mathcal{H}_3^{j-1}$ and $\mathcal{H}_3^j$ are distinguishable then we construct and adversary $\mathcal{A}^{\mathcal{F}}$ that distinguishes between the output of $\mathsf{hc}(x)$ and a random string of λ bits having as input $f^\lambda(x)$. Consider an execution where $R^\star_{\mathcal{OT}}$ has non-negligible advantage in distinguishing $\mathcal{H}_3^{j-1}$ from $\mathcal{H}_3^j$ and consider the randomness ρ used by $R^\star_{\mathcal{OT}}$ and the first round computed by $R^\star_{\mathcal{OT}}$ in this execution, let us say $\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1$. $\mathcal{A}^{\mathcal{F}}$, on input the randomness ρ, the message $b^\star$ committed in $\mathtt{com}$ and the message $r_{1-b^\star}^1$ committed $\mathtt{tcom}_{1-b^\star}$, executes the following steps.

1. Start $R^\star_{\mathcal{OT}}$ with randomness ρ.
2. Let $f, H, f^\lambda(x)$ be the challenge. Upon receiving the first round $(\mathtt{com}, \mathtt{tcom}_0, \mathtt{tcom}_1)$ by $R^\star_{\mathcal{OT}}$, compute $R_{1-b^\star} = r_{1-b^\star}^1 \oplus f^\lambda(x)$, pick a random string $R_{b^\star}$, compute $(f_{0,1}, f_{0,1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $(f_{1,1}, f_{1,1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ send $R_0, R_1, f_{0,1}, f_{1,1}$ to $R^\star_{\mathcal{OT}}$.
3. Continue the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}_3^{j-1}$ using $f_{1-b^\star,j} = f$ instead of using the generation function $\mathsf{Gen}(\cdot)$ when it is required.
4. Upon receiving $(\mathtt{tdec}_0^j, r_0^j), (\mathtt{tdec}_1^j, r_1^j)$ compute $W_{1-b^\star}^j = l_{1-b^\star} \oplus H$,[12] $W_{b^\star}^j = l_{b^\star} \oplus \mathsf{hc}(f_{b^\star,j}^{-\lambda}(r_{b^\star}^j \oplus R_{b^\star}))$, $(f_{0,j+1}, f_{0,j+1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$, $(f_{1,j+1}, f_{1,j+1}^{-1}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends $(W_0^{j+1}, W_1^{j+1}, f_{0,j+1}, f_{1,j+1})$.
5. Continue the interaction with $R^\star_{\mathcal{OT}}$ according to $\mathcal{H}_3^{j-1}$ (and $\mathcal{H}_3^j$) and output what $R^\star_{\mathcal{OT}}$ outputs.

This step of the security proof ends with the observation that if $H = \mathsf{hc}(x)$ then $R^\star_{\mathcal{OT}}$ acts as in $\mathcal{H}_3^{j-1}$, otherwise he acts as in $\mathcal{H}_3^j$.

The second step of the security proof is almost identical to the proof used to argue the indistinguishability between the outputs of $\mathcal{H}_0$ and $\mathcal{H}_2$.

The entire security proof is almost over, indeed the output of $\mathcal{H}_3^\gamma$ corresponds to the output of the simulator Sim and $\mathsf{OUT}_{\mathcal{H}_3, R^\star_{\mathcal{OT}}(z)}(1^\lambda) = \mathsf{OUT}_{\mathcal{H}_3^1, R^\star_{\mathcal{OT}}(z)}(1^\lambda) \approx \mathsf{OUT}_{\mathcal{H}_3^2, R^\star_{\mathcal{OT}}(z)}(1^\lambda) \cdots \approx \mathsf{OUT}_{\mathcal{H}_3^\gamma, R^\star_{\mathcal{OT}}(z)}(1^\lambda)$. Therefore we can claim that the output of $\mathcal{H}_0$ is indistinguishable from the output of Sim when at most one between l_0 and l_1 is used.

$\square$

Theorem 2. *Assuming TDPs, for any $\gamma > 0$ $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ securely computes $F^m_{\mathcal{OT}}$ with one-sided simulation. Moreover the third round is* replayable.

Proof. The third round of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ is *replayable* due to the same arguments used in the security proof of Theorem 1. We now prove that $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ securely computes $F^m_{\mathcal{OT}}$ with one-sided simulation according to Definition 13. More precisely to prove the security against the malicious sender $S^\star_{\overrightarrow{\mathcal{OT}}}$ we start by consider the execution $\mathcal{H}_0$ that correspond to the real execution where the input $b_1, \dots, b_m$ is used by the receiver and then we consider the experiment $\mathcal{H}_i$ where the input used by the receiver is $1 - b_1, \dots, 1 - b_i, b_{i+1}, \dots, b_m$. Suppose now by contradiction that the output

[12]It is important to observe that $r_{b^\star}^1 = r_{b^\star}^j$.

distributions of $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ (for some $i \in \{1, m-1\}$) are distinguishable, then we can construct a malicious sender $S^{\star}_{\mathcal{OT}}$ that breaks the security of $\Pi^{\gamma}_{\mathcal{OT}}$ against malicious sender. This allow us to claim that the output distribution of $\mathcal{H}_0$ is indistinguishable from the output distribution of $\mathcal{H}_m$. A similar proof can be made when the malicious party is the receiver. More precisely, we start by consider the execution $\mathcal{H}_0$ that correspond to the real execution where the input $((l_0^1, l_1^1), \dots, (l_0^m, l_1^m))$ is used by the sender and then we consider the experiment $\mathcal{H}_i$ where the simulator instead of the honest sender procedure is used in the first i parallel executions of $\Pi^{\gamma}_{\mathcal{OT}}$. Supposing by contradiction that the output distributions of $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ (for some $i \in \{1, m-1\}$) are distinguishable, then we can construct a malicious receiver $R^{\star}_{\mathcal{OT}}$ that breaks the security of $\Pi^{\gamma}_{\mathcal{OT}}$ against malicious sender. We observe that in $\mathcal{H}_i$ in the first i parallel executions of $\Pi^{\gamma}_{\mathcal{OT}}$ the simulator Sim is used and this could disturb the reduction to the security of $\Pi^{\gamma}_{\mathcal{OT}}$ when proving that the output distribution of $\mathcal{H}_i$ is indistinguishable from the output distribution of $\mathcal{H}_{i+1}$. In order to conclude the security proof we need just to show that Sim's behaviour does not disturb the reduction. As described in the security proof of $\Pi^{\gamma}_{\mathcal{OT}}$, the simulation made by Sim roughly works by rewinding from the third to the second round while from the fourth round onwards Sim works straight line. An important feature enjoyed by Sim is that he maintains the main thread. Let $\mathcal{C}^{\mathcal{OT}}$ be the challenger of $\Pi^{\gamma}_{\mathcal{OT}}$ against malicious receiver, our adversary $R^{\star}_{\mathcal{OT}}$ works as following.

1. Upon receiving the first round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ from $R^{\star}_{\overrightarrow{\mathcal{OT}}}$, forward the $(i+1)$-th component ot_1 to $\mathcal{C}^{\mathcal{OT}}$[13].
2. Upon receiving ot_2 from $\mathcal{C}^{\mathcal{OT}}$ interacts against $R^{\star}_{\overrightarrow{\mathcal{OT}}}$ by computing the second round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ according to $\mathcal{H}_i$ ($\mathcal{H}_{i+1}$) with the difference that in the $(i+1)$-th position the value ot_2 is used.
3. Upon receiving the third round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ from $R^{\star}_{\overrightarrow{\mathcal{OT}}}$, forward the $(i+1)$-th component ot_3 to $\mathcal{C}^{\mathcal{OT}}$.
4. Upon receiving ot_4 from $\mathcal{C}^{\mathcal{OT}}$ interacts against $R^{\star}_{\overrightarrow{\mathcal{OT}}}$ by computing the fourth round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ according to $\mathcal{H}_i$ ($\mathcal{H}_{i+1}$) with the difference that in the $(i+1)$-th position the value ot_4 is used.
5. for $i = 2, \dots, \gamma$ follow the strategy described in step 3 and 4 and output what $R^{\star}_{\overrightarrow{\mathcal{OT}}}$ outputs.

We recall that in $\mathcal{H}_i$ (as well as in $\mathcal{H}_{i+1}$) in the first i execution of $\Pi^{\gamma}_{\mathcal{OT}}$ the simulator is used, therefore a rewind is made from the third to the second round. During the rewinds $R^{\star}_{\mathcal{OT}}$ can forward to $R^{\star}_{\overrightarrow{\mathcal{OT}}}$ the same second round ot_2. Moreover, due to the main thread property enjoyed by Sim, after the rewind $R^{\star}_{\mathcal{OT}}$ can continue the interaction against $R^{\star}_{\overrightarrow{\mathcal{OT}}}$ without rewind $\mathcal{C}^{\star}$. Indeed if Sim does not maintains the main thread then, even though the same ot_2 is used during the rewind, $R^{\star}_{\overrightarrow{\mathcal{OT}}}$ could send a different ot_3 making impossible to efficiently continue the reduction.

□

3.5 Secure 2PC in the Simultaneous Message Exchange Model

Overview of our protocol: $\Pi_{2\mathcal{PC}} = (P_1, P_2)$. In this section we give an high-level overview of our 4-round 2PC protocol $\Pi_{2\mathcal{PC}} = (P_1, P_2)$ for every functionality $F = (F_1, F_2)$ in the simultaneous message exchange model. $\Pi_{2\mathcal{PC}}$ consists of two simultaneous symmetric executions

[13]We recall that $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ is constructed by executing in parallel m instantiations of $\Pi^{\gamma}_{\mathcal{OT}}$, therefore in this reduction we are just replacing the $(i+1)$-th component of every rounds sent to $R^{\star}_{\overrightarrow{\mathcal{OT}}}$ with the value received by $\mathcal{C}^{\mathcal{OT}}$. Vice versa, we forward to $\mathcal{C}^{\star}$ the $(i+1)$-th component of the rounds received from $R^{\star}_{\overrightarrow{\mathcal{OT}}}$.

of the same subprotocol in which only one party learns the output. In the rest of the thesis we indicate as left execution the execution of the protocol where P_1 learns the output and as right execution the execution of the protocol where P_2 learns the output. In Fig. 3.6 we provide the high level description of the left execution of $\Pi_{2\mathcal{PC}}$. We denoted by (m_1, m_2, m_3, m_4) the messages played in the left execution where (m_1, m_3) are sent by P_1 and (m_2, m_4) are sent by P_2. Likewise, in the right execution of the protocol the messages are denoted by $(\tilde{m}_1, \tilde{m}_2, \tilde{m}_3, \tilde{m}_4)$ where $(\tilde{m}_1, \tilde{m}_3)$ are sent by P_2 and $(\tilde{m}_2, \tilde{m}_4)$ are sent by P_1. Therefore, messages $(m_j, \tilde{m}_j)$ are exchanged simultaneously in the j-th round, for $j \in \{1, \ldots, 4\}$. Our construction uses the following tools.

- A non-interactive perfectly binding computationally hiding commitment scheme $\mathsf{PBCOM} = (\mathsf{Com}, \mathsf{Dec})$.
- A Yao's garbled circuit scheme $(\mathsf{GenGC}, \mathsf{EvalGC})$ with simulator SimGC.
- A protocol $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}} = (S_{\overrightarrow{\mathcal{OT}}}, R_{\overrightarrow{\mathcal{OT}}})$ that securely computes $F^{m}_{\mathcal{OT}}$ with one-sided simulation.
- A 4-round delayed-input NMZK AoK $\mathsf{NMZK} = (\mathcal{P}_{\mathsf{NMZK}}, \mathcal{V}_{\mathsf{NMZK}})$ for the $\mathcal{NP}$-language L_{NMZK} that will be specified later (see Sec. 3.5.1 for the formal definition of L_{NMZK}).

In Figure 3.6 we propose the high-level description of the left execution of $\Pi_{2\mathcal{PC}}$ where P_1 runs on input $x \in \{0,1\}^{\lambda}$ and P_2 on input $y \in \{0,1\}^{\lambda}$.

3.5.1 Formal Description of Our $\Pi_{2\mathcal{PC}} = (P_1, P_2)$

We first start by defining the following $\mathcal{NP}$-language

$$\begin{aligned}
L_{\mathsf{NMZK}} = \big\{ \big(\mathtt{com}_{\mathsf{GC}}, \mathtt{com}_{\mathsf{L}}, \mathsf{GC}, (\mathsf{ot}^1, \mathsf{ot}^2, \mathsf{ot}^3, \mathsf{ot}^4)\big) :\\
\exists(\mathtt{dec}_{\mathsf{GC}}, \mathtt{dec}_{\mathsf{L}}, \mathsf{input}, \alpha, \beta, \omega) \text{ s.t.}\\
\big((Z_{1,0}, Z_{1,1}, \ldots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC}) \leftarrow \mathsf{GenGC}(1^{\lambda}, F_1, \mathsf{input}; \omega)\big) \text{ AND}\\
\big(\mathsf{Dec}(\mathtt{com}_{\mathsf{L}}, \mathtt{dec}_{\mathsf{L}}, Z_{1,0}||Z_{1,1}||, \ldots, ||Z_{\lambda,0}||Z_{\lambda,1}) = 1\big) \text{ AND}\\
\big(\mathsf{ot}^1 \text{ and } \mathsf{ot}^3 \text{are obtained by running } R_{\overrightarrow{\mathcal{OT}}} \text{ on input } 1^{\lambda}, \mathsf{input}, \alpha\big) \text{ AND}\\
\big(\tilde{\mathsf{ot}}^2 \text{ and } \tilde{\mathsf{ot}}^4 \text{ are obtained by running } S_{\overrightarrow{\mathcal{OT}}} \text{ on input}\\
(1^{\lambda}, Z_{1,0}, Z_{1,1}, \ldots, Z_{\lambda,0}, Z_{\lambda,1}, \beta)\big)\big\}.
\end{aligned}$$

The NMZK AoK NMZK used in our protocol is for the $\mathcal{NP}$-language L_{NMZK} described above. Now we are ready to describe our protocol $\Pi_{2\mathcal{PC}} = (P_1, P_2)$ in a formal way.

Protocol $\Pi_{2\mathcal{PC}} = (P_1, P_2)$

Common input: security parameter λ and instance length ℓ_{NMZK} of the statement of the NMZK.

P_1's input: $x \in \{0,1\}^{\lambda}$, *P_2's input:* $y \in \{0,1\}^{\lambda}$.

Round 1. In this round P_1 sends the message m_1 and P_2 the message $\tilde{m}_1$. The steps computed by P_1 to construct m_1 are the following.

1. Run $\mathcal{V}_{\mathsf{NMZK}}$ on input the security parameter 1^{λ} and ℓ_{NMZK} thus obtaining the first round nmzk^1 of NMZK.
2. Run $R_{\overrightarrow{\mathcal{OT}}}$ on input 1^{λ}, x and the randomness α thus obtaining the first round ot^1 of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$.
3. Set $m_1 = \big(\mathsf{nmzk}^1, \mathsf{ot}^1\big)$ and send m_1 to P_2.

Likewise, P_2 performs the same actions of P_1 constructing message $\tilde{m}_1 = \big(\tilde{\mathsf{nmzk}}^1, \tilde{\mathsf{ot}}^1\big)$.

Round 2. In this round P_2 sends the message m_2 and P_1 the message $\tilde{m}_2$. The steps computed by P_2 to construct m_2 are the following.

1. Compute $(Z_{1,0}, Z_{1,1}, \ldots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC_y}) \leftarrow \mathsf{GenGC}(1^{\lambda}, F_2, y; \omega)$.

$P_1(x)$ $P_2(y)$

Run $R_{\overrightarrow{\mathcal{OT}}}$ on input $1^\lambda, x$ and randomness α to get ot^1;
Run $\mathcal{V}_{\mathsf{NMZK}}$ on input 1^λ to get nmzk^1.

$m_1 = (\mathsf{ot}^1, \mathsf{nmzk}^1)$
$\longrightarrow$

$(Z_{1,0}, Z_{1,1}, \dots, Z_{\lambda,0}, Z_{\lambda,1}, \mathsf{GC_y})$
$\leftarrow \mathsf{GenGC}(1^\lambda, F_1, y; \omega)$;
$(\mathsf{com_L}, \mathsf{dec_L})$
$\leftarrow \mathsf{Com}(Z_{1,0}||Z_{1,1}||, \dots, ||Z_{\lambda,1})$;
$(\mathsf{com_{GC_y}}, \mathsf{dec_{GC_y}}) \leftarrow \mathsf{Com}(\mathsf{GC_y})$;
Run $S_{\overrightarrow{\mathcal{OT}}}$ on input $1^\lambda, \mathsf{ot}^1$, $(Z_{1,0}, Z_{1,1}, \dots, Z_{\lambda,0}, Z_{\lambda,1})$
and randomness β to get ot^2;
Run $\mathcal{P}_{\mathsf{NMZK}}$ on input 1^λ and nmzk^1 to get nmzk^2.

$m_2 = (\mathsf{com_L}, \mathsf{ot}^2, \mathsf{com_{GC_y}}, \mathsf{nmzk}^2)$
$\longleftarrow$

Run $R_{\overrightarrow{\mathcal{OT}}}$ on input ot^2 to get ot^3;
Run $\mathcal{V}_{\mathsf{NMZK}}$ on input nmzk^2 to get nmzk^3.

$m_3 = (\mathsf{c}_0, \mathsf{z}_0, \mathsf{c}_1, \mathsf{z}_1, \mathsf{ot}^3, \mathsf{nmzk}^3)$
$\longrightarrow$

Run $S_{\overrightarrow{\mathcal{OT}}}$ on input ot^3 to get ot^4;
Run $\mathcal{P}_{\mathsf{NMZK}}$ on input nmzk^3, stm[a] and w_{stm}[b] to get nmzk^4.

$m_4 = (\mathsf{ot}^4, \mathsf{GC_y}, \mathsf{nmzk}^4)$
$\longleftarrow$

Run $R_{\overrightarrow{\mathcal{OT}}}$ in input ot^4 thus obtaining $Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda}$;
If $\mathcal{V}_{\mathsf{NMZK}}$ on input nmzk^4 and stm outputs 1
  output $v = \mathsf{EvalGC}(\mathsf{GC_y}, Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda})$;
otherwise output $\perp$.

[a]Informally, NMZK proves that: 1) P_2 has performed both oblivious transfers correctly using the same input y; 2) the Yao's gabled circuit $\mathsf{GC_y}$ is correctly computed and 3) the value $\mathsf{GC_y}$ sent in the last round represents the message committed in $\mathsf{com_{GC_y}}$.

[b]w_{stm} is s.t. $(\mathsf{stm}, w_{\mathsf{stm}}) \in \mathsf{Rel}_{\mathsf{L_{NMZK}}}$.

Figure 3.6: High-level description of the left execution of $\Pi_{2\mathcal{PC}}$.

2. Compute $(\mathtt{com}_{\mathsf{GC_y}}, \mathtt{dec}_{\mathsf{GC_y}}) \leftarrow \mathsf{Com}(\mathsf{GC_y})$ and $(\mathtt{com}_{\mathsf{L}}, \mathtt{dec}_{\mathsf{L}}) \leftarrow \mathsf{Com}(Z_{1,0}||Z_{1,1}||, \dots, ||Z_{\lambda,0}||Z_{\lambda,1})$[14].
3. Run $\mathcal{P}_{\mathsf{NMZK}}$ on input 1^λ and nmzk^1 thus obtaining the second round nmzk^2 of NMZK.
4. Run $S_{\overrightarrow{\mathcal{OT}}}$ on input $1^\lambda, Z_{1,0}, Z_{1,1}, \dots, Z_{\lambda,0}, Z_{\lambda,1}$, ot^1 and the randomness β thus obtaining the second round ot^2 of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$.
5. Set $m_2 = \left(\mathsf{ot}^2, \mathtt{com}_{\mathsf{L}}, \mathtt{com}_{\mathsf{GC_y}}, \mathsf{nmzk}^2\right)$ and send m_2 to P_1.

Likewise, P_2 performs the same actions of P_1 constructing message $\tilde{m}_2 = \left(\tilde{\mathsf{ot}}^2, \tilde{\mathtt{com}}_{\mathsf{L}}, \tilde{\mathtt{com}}_{\tilde{\mathsf{GC}}_x}, \tilde{\mathsf{nmzk}}^2\right)$.

Round 3. In this round P_1 sends the message m_3 and P_2 the message $\tilde{m}_3$. The steps computed by P_1 to construct m_3 are the following.

1. Run $\mathcal{V}_{\mathsf{NMZK}}$ on input nmzk^2 thus obtaining the third round nmzk^3 of NMZK.
2. Run $R_{\overrightarrow{\mathcal{OT}}}$ on input ot^2 thus obtaining the third round ot^3 of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$.
3. Set $m_3 = \left(\mathsf{nmzk}^3, \mathsf{ot}^3\right)$ and send m_3 to P_2.

Likewise, P_2 performs the same actions of P_1 constructing message $\tilde{m}_3 = \left(\tilde{\mathsf{nmzk}}^3, \tilde{\mathsf{ot}}^3\right)$.

Round 4. In this round P_2 sends the message m_4 and P_1 the message $\tilde{m}_4$. The steps computed by P_2 to construct m_4 are the following.

1. Run $S_{\overrightarrow{\mathcal{OT}}}$ on input ot^3, thus obtaining the fourth round ot^4 of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$.
2. Set $\mathtt{stm} = (\mathtt{com}_{\mathsf{GC_y}}, \mathtt{com}_{\mathsf{L}}, \mathsf{GC_y}, \tilde{\mathsf{ot}}_1, \mathsf{ot}_2, \tilde{\mathsf{ot}}_3, \mathsf{ot}_4)$ and $w_{\mathtt{stm}} = (\mathtt{dec}_{\mathsf{GC_y}}, \mathtt{dec}_{\mathsf{L}}, y, \tilde{\alpha}, \beta, \omega)$.
3. Run $\mathcal{P}_{\mathsf{NMZK}}$ on input nmzk^3, $\mathtt{stm}$ and $w_{\mathtt{stm}}$ thus obtaining the fourth round nmzk^4 of NMZK.
4. Set $m_4 = \left(\mathsf{nmzk}^4, \mathsf{ot}^4, \mathsf{GC_y}\right)$ and send m_4 to P_1.

Likewise, P_1 performs the same actions of P_2 constructing message $\tilde{m}_4 = \left(\tilde{\mathsf{nmzk}}^4, \tilde{\mathsf{ot}}^4, \tilde{\mathsf{GC}}_x\right)$.

Output computation.

P_1**'s output:** P_1 checks if the transcript $(\mathsf{nmzk}^1, \mathsf{nmzk}^2, \mathsf{nmzk}^3, \mathsf{nmzk}^4)$ is accepting w.r.t. $\mathtt{stm}$. In the negative case P_1 outputs $\perp$, otherwise P_1 runs $R_{\overrightarrow{\mathcal{OT}}}$ on input ot^4 thus obtaining $Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda}$ and computes the output $v_1 = \mathsf{EvalGC}(\mathsf{GC_y}, Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda})$.

P_2**'s output:** P_2 checks if the transcript $\tilde{\mathsf{nmzk}}^1, \tilde{\mathsf{nmzk}}^2, \tilde{\mathsf{nmzk}}^3, \tilde{\mathsf{nmzk}}^4$ is accepting w.r.t. $\tilde{\mathtt{stm}}$. In the negative case P_2 outputs $\perp$, otherwise P_2 runs $R_{\overrightarrow{\mathcal{OT}}}$ on input $\tilde{\mathsf{ot}}^4$ thus obtaining $\tilde{Z}_{1,y_1}, \dots, \tilde{Z}_{\lambda,y_\lambda}$ and computes the output $v_2 = \mathsf{EvalGC}(\tilde{\mathsf{GC}}_x, \tilde{Z}_{1,y_1}, \dots, \tilde{Z}_{\lambda,y_\lambda})$.

High-level overview of the security proof. Due to the symmetrical nature of the protocol, it is sufficient to prove the security against one party (let this party be P_2). We start with the description of the simulator Sim. Sim starts the simulator of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ $\mathsf{Sim}_{\mathcal{OT}}$ which outputs the input $y^\star$ used by the malicious party. Sim sends $y^\star$ to the ideal functionality F and receives back $(v_1, v_2) = \left(F_1(x, y^\star), F_2(x, y^\star)\right)$. Then, Sim computes $(\tilde{\mathsf{GC}}_\star, (\tilde{Z}_1, \dots, \tilde{Z}_\lambda)) \leftarrow \mathsf{SimGC}(1^\lambda, F_2, y^\star, v_2)$ and answer to $\mathsf{Sim}_{\mathcal{OT}}$ using $(\tilde{Z}_1, \dots, \tilde{Z}_\lambda)$ (we recall that Sim acts as the ideal functionality $F^m_{\mathcal{OT}}$ for $\mathsf{Sim}_{\mathcal{OT}}$). Moreover, instead of committing to the labels of Yao's garbled circuit, in the second round Sim commits to 0 and $\tilde{\mathsf{GC}}_\star$ is sent in the last round. Then Sim runs the simulator $\mathsf{Sim}_{\mathsf{NMZK}}$ of NMZK. For the messages of $\Pi_{\mathcal{OT}}$ where P_1 acts as the receiver, Sim runs $R_{\overrightarrow{\mathcal{OT}}}$ on input 0^λ instead of using x. In our security proof we proceed through a sequence of hybrid experiments, where the first one corresponds to the real-world execution and the final represents the execution of Sim in the ideal world. The core idea of our approach is to run the simulator of NMZK, while extracting the input from $P_2^\star$. By running the simulator of NMZK we are able to guarantee that the value extracted via $\mathsf{Sim}_{\mathcal{OT}}$ is correct, even though $P_2^\star$ is receiving proofs for a false statement. Indeed in each intermediate hybrid experiment that we will consider, also the extractor of NMZK is run in order to extract the witness for the theorem proved by $P_2^\star$. In this way we can prove that the value extracted via $\mathsf{Sim}_{\mathcal{OT}}$ is consistent with

[14]Instead of one commitment for each label, P_2 commits to the concatenation of all the labels of the garbled circuit $\mathsf{GC_y}$.

the input that $P_2^\star$ is using in the other part of the protocol (e.g. in the construction of the garbled circuit and in the execution of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ in which the adversary acts as a sender). For what we have discussed, the simulator of NMZK rewinds first from the third to the second round (to extract the trapdoor), and then from the fourth to the third round (to extract the witness for the statement proved by $P_2^\star$). We need to show that these rewinding procedures do not disturb the security proof when we rely on the security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. This is roughly the reason why we require the third round of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ to be reusable and the security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ against malicious receiver to hold for any $\gamma = \mathsf{poly}(\lambda)$.

Theorem 3. *Assuming TDPs, $\Pi_{2\mathcal{PC}}$ securely computes every two-party functionality $F = (F_1, F_2)$ with black-box simulation.*

Proof. In order to prove that $\Pi_{2\mathcal{PC}}$ securely computes $F = (F_1, F_2)$, we first observe that, due to the symmetrical nature of the protocol, it is sufficient to prove the security against one party (let this party be P_2). We now show that for every adversary $P_2^\star$, there exists an ideal-world adversary (simulator) Sim such that for all inputs x, y of equal length and security parameter λ:

$$\{\mathsf{REAL}_{\Pi_{2\mathcal{PC}}, P_2^\star(z)}(1^\lambda, x, y)\} \approx \{\mathsf{IDEAL}_{F,\mathsf{Sim}(z)}(1^\lambda, x, y)\}.$$

Our simulator Sim is the one showed in Sec. 3.5.1.

In our security proof we proceed through a series of hybrid experiments, where the first one corresponds to the execution of $\Pi_{2\mathcal{PC}}$ between P_1 and $P_2^\star$ (real-world execution). Then, we gradually modify this hybrid experiment until the input of the honest party is not needed anymore, such that the final hybrid would represent the simulator (simulated execution).

We now give the descriptions of the hybrid experiments and of the corresponding security reductions. We denote the output of $P_2^\star$ and the output of the procedure that interacts against $P_2^\star$ on the behalf of P_1 in the hybrid experiment $\mathcal{H}_i$ with $\{\mathsf{OUT}_{\mathcal{H}_i, P_2^\star(z)}(1^\lambda, x, y)\}_{x\in\{0,1\}^\lambda, y\in\{0,1\}^\lambda}$.

- $\mathcal{H}_0$ corresponds to the real executions. More in details, $\mathcal{H}_0$ runs $P_2^\star$ with a fresh randomness, and interacts with it as the honest player P_1 does using x as input. The output of the experiment is $P_2^\star$'s view and the output of P_1. Note that we are guarantee from the soundness of NMZK that $\mathsf{stm} \in L_{\mathsf{NMZK}}$, that is: 1)$P_2^\star$ uses the same input $y^\star$ in both the OT executions; 2) the garbled circuit committed in $\mathtt{com}_{\mathsf{GC_y}}$ and the corresponding labels committed in $\mathtt{com}_{\mathsf{L}}$ are computed using the input $y^\star$; 3) the garbled circuit sent in the last round is actually the one committed in $\mathtt{com}_{\mathsf{GC_y}}$.
- $\mathcal{H}_1$ proceeds in the same way of $\mathcal{H}_0$ except that the simulator $\mathsf{Sim}_{\mathsf{NMZK}}$ of NMZK is used in order to compute the messages of NMZK played by P_1. Note that $\mathsf{Sim}_{\mathsf{NMZK}}$ rewinds $P_2^\star$ from the 3rd to the 2nd round in oder to extract the trapdoor. The indistinguishability between the output distribution of these two hybrids experiments holds from the property 1 of NMZK (see Definition 9). In this, and also in the next hybrids, we prove that $\text{Prob}\left[\, \mathsf{stm} \notin L_{\mathsf{NMZK}} \,\right] \leq \nu(\lambda)$. That is, we prove that $P_2^\star$ behaves honestly across the hybrid experiments even though he is receiving a simulated proof w.r.t. NMZK and $\tilde{\mathsf{stm}}$ does not belong to L_{NMZK}. In this hybrid experiment we can prove that if by contradiction this probability is non-negligible, then we can construct a reduction that breaks the property 2 of NMZK (see Definition 9). Indeed, in this hybrid experiment, the theorem that $P_2^\star$ receives belongs to L_{NMZK} and the simulator of $\mathsf{Sim}_{\mathsf{NMZK}}$ is used in order to compute and accepting transcript w.r.t. NMZK. Therefore, relying on property 2 of Definition 9 we know that there exists a simulator that extracts the the witness for the statement stm proved by $P_2^\star$ with all but negligible probability.

- $\mathcal{H}_2$ proceeds in the same way of $\mathcal{H}_1$ except that the simulator of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$, $\mathsf{Sim}_{\mathcal{OT}}$, is used instead of the sender algorithm $S_{\overrightarrow{\mathcal{OT}}}$. We recall that $\mathsf{Sim}_{\mathcal{OT}}$ requires to interact with the ideal functionality $F^m_{\mathcal{OT}}$. In this case the hybrid experiment $\mathcal{H}_2$ acts on the behalf of the $F^m_{\mathcal{OT}}$ by answering to a query $y^\star$ made by $\mathsf{Sim}_{\mathcal{OT}}$ using the garbled circuit labels $(\tilde{Z}_{1,y^\star_1}, \dots, \tilde{Z}_{\lambda,y^\star_\lambda})$. From the simulatable security against malicious receiver of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ for every $\gamma = \mathsf{poly}(\lambda)$ follows that the output distributions of $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable. Suppose by contradiction this claim does not hold, then we can show a malicious receiver $R^\star_{\overrightarrow{\mathcal{OT}}}$ that breaks the simulatable security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ against a malicious receiver. In more details, let $\mathcal{C}_{\mathcal{OT}}$ be the challenger of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. $R^\star_{\overrightarrow{\mathcal{OT}}}$ plays all the messages of $\Pi_{2\mathcal{PC}}$ as in $\mathcal{H}_1$ ($\mathcal{H}_2$) except for the messages of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. For these messages $R^\star_{\overrightarrow{\mathcal{OT}}}$ acts as a proxy between $\mathcal{C}_{\mathcal{OT}}$ and $P_2^\star$. In the end of the execution $R^\star_{\overrightarrow{\mathcal{OT}}}$ runs the distinguisher D that distinguishes $\{\mathsf{OUT}_{\mathcal{H}_1,P_2^\star(z)}(1^\lambda, x, y)\}$ from $\{\mathsf{OUT}_{\mathcal{H}_2,P_2^\star(z)}(1^\lambda, x, y)\}$ and outputs what D outputs. We observe that if $\mathcal{C}_{\mathcal{OT}}$ acts as the simulator then $P_2^\star$ acts as in $\mathcal{H}_2$ otherwise he acts as in $\mathcal{H}_1$.

  To argue that Prob $[\,\mathsf{stm} \notin L_{\mathsf{NMZK}}\,] \le \nu(\lambda)$ also in this hybrid experiment we use the simulator-extractor $\mathsf{Sim}_{\mathsf{NMZK}}$ in order to check whether the theorem proved by $P_2^\star$ is still true. If it is not the case then we can construct a reduction to the simulatable security against malicious receiver of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$. Note that $\mathsf{Sim}_{\mathsf{NMZK}}$ rewinds from the 4th to the 3rd round in order to extract the witness $w_{\mathtt{stm}}$ for the statement $\mathtt{stm}$ proved by $P_2^\star$. These rewinds could cause $P_2^\star$ to ask multiple third rounds of OT $\tilde{\mathsf{ot}}^3_i$ ($i = 1, \dots, \mathsf{poly}(\lambda)$). In this case $R^\star_{\overrightarrow{\mathcal{OT}}}$ can simply forward $\tilde{\mathsf{ot}}^3_i$ to $\mathcal{C}_{\mathcal{OT}}$ and obtains from $\mathcal{C}_{\mathcal{OT}}$ an additional $\tilde{\mathsf{ot}}^4_i$. This behaviour of $R^\star_{\overrightarrow{\mathcal{OT}}}$ is allowed because $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ is simulatable secure against a malicious receiver even when the last two rounds of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ are executed γ times (as stated in Theorem 1). Therefore the reduction still works if we set γ equals to the expected number of rewinds that $\mathsf{Sim}_{\mathsf{NMZK}}$ could do. We observe that since we have proved that $\mathsf{stm} \in L_{\mathsf{NMZK}}$, then the value $y^\star$ queried by $\mathsf{Sim}_{\mathcal{OT}}$ corresponds to the input used by $P_2^\star$ in the overall execution of the protocol. That is, $P_2^\star$ uses $y^\star$ to both compute the garbled circuit and to complete the execution $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ in which she acts as a sender. On top of this observation, we obtain that in $\mathcal{H}_1$ the value $v_1 = F_1(x, y^\star)$ corresponds, with overwhelming probability, to the valued computed by running the garbled circuit $\mathsf{GC_y}$ received by $P_2^\star$ using as input the labels $Z_{1,x_1}, \dots, Z_{\lambda,x_\lambda}$.
- $\mathcal{H}_3$ differs from $\mathcal{H}_2$ in the way the rounds of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$, where $P_2^\star$ acts as sender, are computed. More precisely instead of using x as input, 0^λ is used. Note that from this hybrid onward it is not possible anymore to compute the output by running EvalGC as in the previous hybrid experiments. This is because we are not able to recover the correct labels to evaluate the garbled circuit. Therefore $\mathcal{H}_3$ computes the output by directly evaluating $v_1 = F_1(x, y^\star)$, where $y^\star$ is the input of $P_2^\star$ obtained by $\mathsf{Sim}_{\mathcal{OT}}$.

  The indistinguishability between the output distributions of $\mathcal{H}_3$ and $\mathcal{H}_2$ comes from the security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ against malicious sender. Indeed, suppose by contradiction that it is not the case, then we can show a malicious sender $S^\star_{\overrightarrow{\mathcal{OT}}}$ that breaks the indistinguishability based security of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ against a malicious sender. In more details, let $\mathcal{C}_{\mathcal{OT}}$ be the challenger. $S^\star_{\overrightarrow{\mathcal{OT}}}$ plays all the messages of $\Pi_{2\mathcal{PC}}$ as in $\mathcal{H}_3$ ($\mathcal{H}_2$) except for the messages of OT where he acts as a receiver. For these messages $S^\star_{\overrightarrow{\mathcal{OT}}}$ plays as a proxy between $\mathcal{C}_{\mathcal{OT}}$ and $P_2^\star$. At the end of the execution $S^\star_{\overrightarrow{\mathcal{OT}}}$ runs the distinguisher D that distinguishes $\{\mathsf{OUT}_{\mathcal{H}_2,P_2^\star(z)}(1^\lambda, x, y)\}$ from $\{\mathsf{OUT}_{\mathcal{H}_3,P_2^\star(z)}(1^\lambda, x, y)\}$ and outputs what D outputs. We observe that if $\mathcal{C}_{\mathcal{OT}}$ computes the messages of $\Pi^{\gamma}_{\overrightarrow{\mathcal{OT}}}$ using the input 0^λ then $P_2^\star$ acts as in $\mathcal{H}_3$ otherwise he acts as in $\mathcal{H}_2$. In this security proof there is another subtlety. During the reduction $S^\star_{\overrightarrow{\mathcal{OT}}}$ runs $\mathsf{Sim}_{\mathsf{NMZK}}$ that rewinds from the third to the second round. This means that $P_2^\star$ could send multiple different second rounds ot_i^2 of OT (with $i = 1, \dots, \mathsf{poly}(\lambda)$). $S^\star_{\overrightarrow{\mathcal{OT}}}$ cannot forward these other messages to $\mathcal{C}_{\mathcal{OT}}$ (he cannot rewind the challenger). This is not a problem because the third round of $\Pi^\gamma_{\overrightarrow{\mathcal{OT}}}$ is replayable (as proved in Theorem 1). That is the round ot^3 received from the challenger can be used to answer to any ot^2. To prove that $\text{Prob}\left[\, \mathsf{stm} \notin L_{\mathsf{NMZK}} \,\right] \leq \nu(\lambda)$ we use the same arguments as before by observing the the rewinds made by the simulator-extractor from the fourth round to the third one do not affect the reduction.

- $\mathcal{H}_4$ proceeds in the same way of $\mathcal{H}_3$ except for the message committed in $\tilde{\mathtt{com}}_{\mathsf{L}}$. More precisely, instead of computing a commitment of the labels $(\tilde{Z}_{1,0}, \tilde{Z}_{1,1}, \dots, \tilde{Z}_{\lambda,0}, \tilde{Z}_{\lambda,1})$, a commitment of $0^\lambda || \dots || 0^\lambda$ is computed. The indistinguishability between the output distributions of $\mathcal{H}_3$ and $\mathcal{H}_4$ follows from the hiding of PBCOM. Moreover, $\text{Prob}\left[\, \mathsf{stm} \notin L_{\mathsf{NMZK}} \,\right] \leq \nu(\lambda)$ in this hybrid experiment due to the same arguments used previously.
- $\mathcal{H}_5$ proceeds in the same way of $\mathcal{H}_4$ except for the message committed in $\tilde{\mathtt{com}}_{\mathsf{GC_y}}$: instead of computing a commitment of the Yao's garbled circuit $\tilde{\mathtt{GC}}_x$, a commitment of 0 is computed. The indistinguishability between the output distributions of $\mathcal{H}_4$ and $\mathcal{H}_5$ follow from the hiding of PBCOM. We have that $\text{Prob}\left[\, \mathsf{stm} \notin L_{\mathsf{NMZK}} \,\right] \leq \nu(\lambda)$ in this hybrid experiment due to the same arguments used previously.
- $\mathcal{H}_6$ proceeds in the same way of $\mathcal{H}_5$ except that the simulator SimGC is run (instead of GenGC) in order to obtain the Yao's garbled circuit and the corresponding labels. In more details, once $y^\star$ is obtained by $\mathsf{Sim}_{\mathcal{OT}}$ (in the third round), the ideal functionality F is invoked on input $y^\star$. Upon receiving $(v_1, v_2) = \big(F_1(x, y^\star), F_2(x, y^\star)\big)$ the hybrid experiment compute $(\tilde{\mathtt{GC}}_\star, \tilde{Z}_1, \dots, \tilde{Z}_\lambda) \leftarrow \mathsf{SimGC}(1^\lambda, F_2, y^\star, v_2)$ and replies to the query made by $\mathsf{Sim}_{\mathcal{OT}}$ with $(\tilde{Z}_1, \dots, \tilde{Z}_\lambda)$. Furthermore, in the 4th round the simulated Yao's garbled circuit $\tilde{\mathtt{GC}}_\star$ is sent, instead of the one generated using GenGC. The indistinguishability between the output distributions of $\mathcal{H}_5$ and $\mathcal{H}_6$ follows from the security of the Yao's garbled circuit. To prove that $\text{Prob}\left[\, \mathsf{stm} \notin L_{\mathsf{NMZK}} \,\right] \leq \nu(\lambda)$ we use the same arguments as before by observing the the rewinds made by the simulator-extractor from the fourth round to the third one do not affect the reduction.

The proof ends with the observation that $\mathcal{H}_6$ corresponds to the simulated execution with the simulator Sim.

□

Chapter 4

Private Set-Membership in the Semi-Honest Setting

4.1 Introduction

Private Set Intersection (PSI) is one of the most practically relevant *secure two-party computation* (2PC) tasks. In PSI two parties hold two sets of strings X and Y, respectively. At the end of the protocol one (or both) party should learn the intersection of the two sets $Z = X \cap Y$ and nothing else about the input of the other party.

There are many real-world applications in which PSI is required. As an example, when mobile users install messaging apps, they need to discover whom among their contacts (from their address book) is also using the app, in order to be able to start communicating seamlessly with them. Doing so requires users to learn the intersection of their contact list with the list of registered users of the service which is stored at the server side. This is typically done by having users send their contact list to the server that can then compute the intersection and return the result to the user. Unfortunately this solution is very problematic not only for the privacy of the user, but for the privacy of the users' contacts as well! In particular, some of the people in the contact list might not want their phone number being transferred and potentially stored by the server, but they have no control over this.[1] Note that this is not just a theoretically interesting problem and that Signal (one of the most popular end-to-end encrypted messaging app) has recently recognized this as being a real problem and offered partial solutions to it.[2] PSI has many other applications, including computing intersections of suspect lists, private matchmaking (comparing interests), *testing human genome* [BBC+11], *privacy-preserving ride-sharing* [HOS17], *botnet detection* [NMH+10], *advertisment conversion rate* [IKN+17] and many more.

History of PSI. From a feasibility point of view, PSI is just a special case of 2PC and therefore any generic 2PC protocol (such as [GMW87]) could be used to securely evaluate PSI instances as well. However, since PSI is a natural functionality that can be applied in numerous real-world applications, many efficient protocols for this specific functionality have been proposed, with early results dating back to the 80s [Sha80, Mea86]. The problem was formally defined

[1]Some apps do not transfer the contact list in cleartext, but a hashed version instead. However, since the domain space of phone numbers is small enough to allow for brute forcing of the hashes, this does not guarantee any real privacy guarantee.

[2]Unfortunately, the Signal team has concluded that current PSI protocols are too inefficient for they application scenario and relied on trusted-hardware instead, in the style of [TLP+17]. See https://signal.org/blog/private-contact-discovery/ for more details on this.

in [FNP04] and follow up work increased the efficiency of PSI protocols to have complexity only *linear* in the inputs of the parties [JL10, CT10]. However, these protocol still require to expensive expensive public-key operations (e.g., exponentiations) for every element in the input sets. As public-key operations are orders of magnitudes more expensive than symmetric key operations, these protocols are not practically efficient for large input sets. In the meanwhile, generic techniques for 2PC had improved by several orders of magnitude and the question of whether special purpose protocols or generic protocols were most efficient has been debated in [HEK12, CT12]. Due to its practical relevance, PSI protocols in the *server-aided* model have been proposed as well [KMRS14].

OT-based PSI. The most efficient PSI protocols today are those following the approach of Phasing [PSSZ15, PSZ14]. These protocols make extensive use of a cryptographic primitive known as *oblivious transfer (OT)*. While OT provably requires expensive public-key operation, OT can be "extended" as shown by [IKNP03, ALSZ13, KK13] i.e., the few necessary expensive public-key operations can be amortized over a very large number of OT instances, and the marginal cost of OT is only a few (faster) symmetric key operations instead. In particular, improvements in OT-extension techniques directly imply improvements to PSI protocols as shown by e.g., [KKRT16, OOS17].

In a nutshell, the Phasing protocol introduced two important novel ideas to the state of the art of PSI. First, they give an efficient instantiation of the *private set membership protocol* (PSM) introduced in [FIPR05] based on OT. Second, they show how to efficiently implement PSI from PSM using hashing techniques. (An overview of their techniques is given below).

Our Contribution. The main contribution of this chapter is to give an efficient instantiation of PSM that provides output in encrypted format and can therefore be combined with further 2PC protocols. Our PSM protocol can be naturally combined with the hashing approach of Phasing to give a PSI protocol with encrypted output achieving the same favourable complexity in the input sizes. This enables to combine the efficiency of modern PSI techniques with the potentials of general 2PC. Combining our protocols with the right 2PC post-processing allows to more efficiently evaluate functionalities of the form $Z = f(X \cap Y)$ for any function f. To the best of our knowledge ours is the best special purpose PSI protocol which can be combined with 2PC techniques for post-processing, and therefore we propose the first alternative to *circuit-based PSI.*

Like in Phasing we only focus on semi-honest security. Using the protocol together with an actively secure OT-extension protocol such as [ALSZ15, KOS15] would result in a protocol with *privacy* but not *correctness* (e.g., the view of the protocol *without* the output can be efficiently simulated), which is a meaningful notion of security in some settings. PSI protocols with security against malicious adversaries have been proposed in e.g., [HL08, RR17]. It is an interesting open problem to design efficient protocols which are both secure against active adversaries and that produce encrypted output. Also, like in Phasing, we only focus on the two-party setting. The recent result of [HV17] has shown that multiparty set-intersection can be computed efficiently. Extending our result to the multiparty case is an interesting future research direction.

We also compare the computation complexity of our scheme for PSM with the circuit-based PSI approaches of [PSZ16]. The result of this comparison is that our protocol has better performance then all the circuit-based PSI approaches (which can be combined with further post-processing) proposed in [PSZ16] (see Sec. 4.4.2 for more details).

4.2 Technical overview

4.2.1 Why phasing and 2PC do not mix

We start with a quick overview of the PSM protocol in Phasing [PSSZ15, PSZ14], to explain why their protocol inherently reveals the intersection to one of the parties. From a high-level point of view, the protocol is conceptually similar to the PSM protocol from OPRF of [FIPR05], except that the OPRF is replaced with a similar functionality efficiently implemented using OT. For simplicity, here we will use the OPRF abstraction.

The goal of a PSM protocol is the following: the receiver R has input x, and the sender S has input a set Y; at the end of the protocol the receiver learns whether $x \in Y$ or not while the sender learns nothing. The protocol starts by using the OPRF subprotocol, so that R learns $x^* = F_k(x)$ (where k is known to S), whereas S learns nothing about x. Now S evaluates the PRF on her own set and sends the set $Y^* = \{y^* = F_k(y) | y \in Y\}$ to R, who checks if $x^* \in Y^*$ and concludes that $x \in Y$ if this is the case. In other words, we map all inputs into pseudorandom strings and then let one of the parties test for membership "in the clear". Since the party performing the test doesn't have access to the mapping (e.g., the PRF key), this party can only check for the membership of x and no other points (i.e., all elements in $Y^* \setminus \{x^*\}$ are indistinguishable from random in R's view).

From the above description, it should be clear that the Phasing PSM inherently reveals the output to one of the parties. Turning this into a protocol which provides encrypted output is a challenging task. Here is an attempt at a "strawman" solutions: we change the protocol such that R still learns the pseudorandom string $x^* = F_k(x)$ corresponding to x, but now S sends a value *for every element in the universe*. Namely, for each i (in the domain of Y) S sends an encryption of whether $i \in Y$ "masked" using $F_k(i)$ e.g., S sends $c_i = F_k(i) \oplus E(i \in Y)$[3]. Now R can compute $c_x \oplus x^* = E(x \in Y)$ i.e., an encrypted version of whether $x \in Y$, which can be then used as input to the next protocol.

While this protocol produces the correct result, its complexity is exponential in the bit-legnth of $|x|$, which is clearly not acceptable.

Intuitively, we know that only a polynomial number of c_i's will contain encryptions of 1, and therefore we need to find a way to "compress" all the c_i corresponding to $i \notin Y$ into a single one, to bring the complexity of the protocol back to $O(|Y|)$. In the following, after defining some useful notation, we give an intuitive explanation on how to do that.

4.2.2 Our protocol

Notation. We introduce some useful (and informal) notation in order to make easier to understand the ideas behind our construction. We let $Y = \{y_1, \dots, y_M\}$ be the input set of S, and we assume w.l.o.g., that $|Y| = M = 2^m$.[4] All strings have the same length e.g., $|x| = |y_i| = \lambda$.[5]. We will use a special symbol $\perp$ such that $x \neq \perp \forall x$. We use a function $\mathsf{Prefix}(x, i)$ that outputs the i most significant bits of x. ($\mathsf{Prefix}(x, i) \neq \mathsf{Prefix}(x, j)$ when $i \neq j$ independently of the value of x) and for simplicity we define $\mathsf{Prefix}(Y, i)$ to be the set constructed by taking the i most significant bits of every element in Y.

[3]The exact format of the "encryption" $E(\cdot)$ would depend on the subsequent 2PC protocol and is irrelevant for this high-level description.

[4]Sets can always be padded with dummy elements, but the complexity of the protocol can match M that in practice can be $M \approx 2^{m-1}$.

[5]We can assume λ to be smaller than the (statistical) security parameter s and we will denote the bit decomposition of x by $x = x_\lambda \dots x_1$.. Otherwise before running the protocol the parties can hash their input down and run the protocol with inputs $h(x)$ and $h(Y) = \{h(y_1), \dots, h(y_M)\}$. Clearly if $x = y_i$ then $h(x) = h(y_i)$, and for correctness we need that $Pr[h(x) \in h(Y) \land x \notin Y] < 2^{-s}$.

The protocol uses a symmetric key encryption scheme $\mathsf{Sym} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with the additional property that given a key $k \leftarrow \mathsf{Gen}(1^s)$ it is possible to efficiently verify if a given ciphertext is in the range of k (see Sec. 4.3 for a formal definition).

Finally, the output of the protocol will be one of two strings γ_0, γ_1 chosen by S, respectively denoting $x \notin Y$ and $x \in Y$. The exact format of the two strings depends on the protocol used for post-processing. For instance if the post-processing protocol is based on: 1) *garbled circuits*, then γ_0, γ_1 will be the labels corresponding to some input wire; 2) *homomorphic encryption*, then $\gamma_b = \mathsf{Enc}(pk, b)$ for some homomorphic encryption scheme Enc; 3) *secret-sharing*, then $\gamma_b = s_2 \oplus b$ where s_2 is a uniformly random share chosen by S, so that if R defines its own share as $s_1 = \gamma_b$ then it holds that $s_1 \oplus s_2 = b$.[6]

In order to "compress" the elements of Y we start by considering an undirected graph with a level structure of $\lambda + 1$ levels. The vertices in the last level of this graph will correspond to the elements of Y. More precisely, we associate the secret key $k_{b_\lambda b_{\lambda-1} \ldots b_1}$ of a symmetric key encryption scheme Sym to each element $y = b_\lambda b_{\lambda-1} \ldots b_1 \in Y$. The main idea is to allow the receiver to obliviously navigate this graph in order to get the key $k_{b_\lambda b_{\lambda-1} \ldots b_1}$ if $x = y$, for some $y = b_\lambda b_{\lambda-1} \ldots b_1 \in Y$, or a special key $k^\star$ otherwise. Moreover we allow the receiver to navigate the graph efficiently, that is, every level of the graph is visited only once.

Once a key k is obtained by the receiver, the sender sends $O(|Y|)$ ciphertexts in a such a way that the key obtained by the receiver can decrypt only one ciphertext. Moreover the plaintext of this ciphertext will correspond to γ_0 or γ_1 depending on whether $x \in Y$ or not.

First step: construct the graph G. Each graph level $i \in \{0, \ldots, \lambda\}$ has size at most $|\mathsf{Prefix}(i, Y)| + 1$.

More precisely, for every $t = b_\lambda b_{\lambda-1} \ldots b_{\lambda-i} \in \mathsf{Prefix}(Y, i)$ there is a node in the level i of G that contains a key $k_{b_\lambda b_{\lambda-1} \ldots b_{\lambda-i}}$. In addition, in the level i there is a special node, called *sink node* that contains a key $k_i^\star$ (which we refer to as *sink key*). The aim of $k_i^\star$ is to represent all the values that do not belong to $\mathsf{Prefix}(i, Y)$.

Let us now describe how the graph G is constructed. First, for $i = 1, \ldots, \lambda$ the key (for a symmetric key encryption scheme) $k_j^\star$ is generated using the generation algorithm $\mathsf{Gen}(\cdot)$. As discussed earlier the aim of this keys is to represent the elements that do not belong to Y. More precisely, the sink key $k_i^\star$, with $i \in \{1, \ldots, \lambda\}$ represents all the values that do not belong to $\mathsf{Prefix}(Y, i)$ and the key $k_\lambda^\star$ (the last sink key) will be used to encrypt the output γ_0 corresponding to non-membership in the last step of our protocol. Note that if $\mathsf{Prefix}(x, i) \notin \mathsf{Prefix}(Y, i)$ then $\mathsf{Prefix}(x, j) \notin \mathsf{Prefix}(Y, j)$ for all $j > i$. Therefore, once entered in a sink node, the *sink path* is never abandoned and thus the final sink key $k_\lambda^\star$, will be retrieved (which allows to recover γ_0).

Let us now give a more formal idea of how G is constructed.

- The root of G is empty, and in the second level there are two vertices k_0 and k_1 where[7], for $b = 0, 1$

$$k_b = \begin{cases} k \leftarrow \mathsf{Gen}(1^s), & \text{if } b \in \mathsf{Prefix}(Y, 1) \\ k_1^\star, & \text{otherwise} \end{cases}$$

- For each vertex k_t in the level $i \in \{1, \ldots, \lambda\}$ and for $b = 0, 1$ create the node $k_{t||b}$ as follows (if it does not exists) and connect k_t to it.

[6]Here we use $\oplus$-secret sharing without loss of generality. Any 2-out-2 secret sharing would work here.
[7]In abuse of notation we refer to a vertex using the key represented by the vertex itself.

$$k_{t||b} = \begin{cases} k \leftarrow \mathsf{Gen}(1^s), & \text{if } t||b \in \mathsf{Prefix}(Y, i+1) \\ k^\star_{i+1}, & \text{if } t||b \notin \mathsf{Prefix}(Y, i+1) \\ k^\star_{i+1}, & \text{if } k_t = k^\star_i \end{cases}$$

We observe that a new node $k_{t||b}$ is generated only when $t||b \in \mathsf{Prefix}(Y, i)$. In the other cases the sink node $k^\star_{i+1}$ is used.

In Fig. 4.1 we show an example of what the graph G looks like for the set $Y = \{010, 011, 110\}$. In this example it is possible to see how, in the 2nd level, all the elements that do not belong to $\mathsf{Prefix}(Y, 2)$ are represented by the sink node $k^\star_2$. Using this technique we have that in the last level of G one node ($k^\star_3$ in this example) is sufficient to represent all the elements that do not belong to Y. Therefore, we have that the last level of G contains at most $|Y| + 1$ elements. We also observe that every level of G cannot contain more than $|Y| + 1$ nodes.

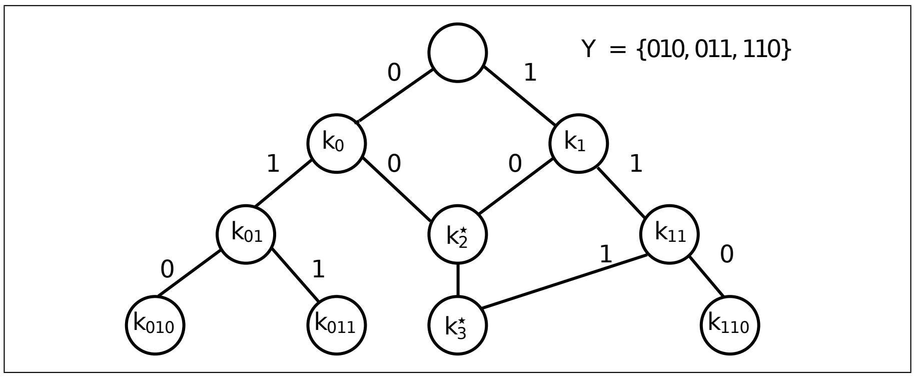


Figure 4.1: Example of how the graph G appears when the sender holds the set Y.

Second step: oblivious navigation of G. Let $x = x_\lambda x_{\lambda-1} \dots x_1$ be the receiver's (R's) private input and Y be the sender's (S's) private input. After S constructs the graph G we need a way to allow R to obtain $k_{x_\lambda x_{\lambda-1} \dots x_1}$ if $x \in Y$ and the sink key $k^\star_\lambda$ otherwise. All the computation has to be done in such a way that no other information about the set Y is leaked to the receiver, and as well that no information about x is leaked to the sender. In order to do so we use λ executions of 1-out-of-2 OT. The main idea is to allow the receiver to select which branch to explore in G depending on the bits of x. More precisely, in the first execution of OT, R will receive the key k_{x_λ} iff there exists an element in Y with the most significant bit equal to x_λ, the sink key $k^\star_1$ otherwise. In the second execution of OT, R uses $x_{\lambda-1}$ as input and S uses (c_0, c_1) where c_0 is computed as follows:

- For each key in the second level of G that has the form $k_{t||0}$, the key $k_{t||0}$ is encrypted using the key k_t.
- For every node v in the first level that is connected to a sink node $k^\star_2$ in the second level, compute an encryption of $k^\star_2$ using the key contained in v.
- Pad the input with random ciphertexts up to the upper bound for the size of this layer (more details about this step are provided later).
- Randomly permute these ciphertexts.

The procedure to compute the input c_1 is essentially the same (the only difference is that in this case we consider every key with form $k_{t||1}$ and encrypt it using k_t).

Roughly, in this step every key contained in a vertex u of the second level is encrypted using the keys contained in the vertex v of the previous level that is connected to u. For example, following the graph provided in Fig.4.1, c_0 would be equal to $\{\mathsf{Enc}(k_0, k_2^\star), \mathsf{Enc}(k_1, k_2^\star)\}$ and c_1 to $\{\mathsf{Enc}(k_0, k_{01}), \mathsf{Enc}(k_1, k_{11})\}$.

Thus, after the execution of OT R receives $c_{x_{\lambda-1}}$ that contains the ciphertexts described above where only one of these can be decrypted using the key k obtained in the first execution of OT. The obtained plaintext corresponds to the key $k_{x_\lambda x_{\lambda-1}}$ if $\mathsf{Prefix}(x, 2) \in \mathsf{Prefix}(Y, 2)$, to the sink key $k_2^\star$ otherwise. The same process is iterated for all the levels of G. More generally if $\mathsf{Prefix}(x, j) \in \mathsf{Prefix}(Y, j)$ then after the $j-$th execution of OT R can compute the key $k_{x_\lambda x_{\lambda-1} \dots x_{\lambda-j}}$ using the key obtained in the previous phase. Conversely if $\mathsf{Prefix}(x, j) \notin \mathsf{Prefix}(Y, j)$ then the sink key $k_j^\star$ is obtained by R. We observe that after every execution of OT R does not know which ciphertext can be decrypted using the key obtained in the previous phase, therefore he will try to decrypt all the ciphertext until the decryption procedure is successful. To avoid adding yet more indexes to the (already heavy) notation of our protocol we deal with this using a private-key encryption scheme with efficiently verifiable range. We note that this is not necessary and that when implementing the protocol one can instead use the *point-and-permute technique* [BMR90]. This, and other optimisations and extensions of our protocol, are described in Section 4.5.

Third step: obtain the correct share. In this step S encrypts the output string γ_0 using the key $k_\lambda^\star$ and uses all the other keys in the last level of G to encrypt the output string γ_1.[8] At this point the receiver can only decrypt either the ciphertext that contains γ_0 if $x \notin Y$ or one (and only one) of the ciphertexts that contain γ_1 if $x \in Y$. In the protocol that we have described so far R does not know which ciphertext can be decrypted using the key that he has obtained. Also in this case we can use a point-and-permute technique to allow R to identify the only ciphertext that can be decrypted using his key.

On the need for padding. As describe earlier, we might need to add some padding to the OT sender's inputs. To see why we need this we made the following observation. We recall that in the i-th OT execution the sender computes an encryption of the keys in the level i of the artificial graph G using the keys of the previous level $(i-1)$.[9] As a result of this computation the sender obtains the couple (c_0^i, c_1^i), that will be used as input of the i-th OT execution, where c_0^i (as well as c_1^i) contains a number of encryptions that depends upon the number of vertices on level $(i-1)$ of G. We observe that this leaks informations about the structure of G to the receiver, and therefore leaks information about the elements that belong to Y. Considering the example in Fig. 4.1, if we allow the receiver to learn that the 2nd level only contains 3 nodes, then the receiver will learn that all the elements of Y have the two most significant bits equal to either t or t' for some $t, t' \in \{0,1\}^2$ (in Fig.4.1 for example we have $t = 01$ and $t' = 11$). However, note that the receiver will not learn the actual values of t and t'.

We finally note that the technique described in this section can be seen as a special (and simpler) example of securely evaluating a branching program. Secure evaluation of branching programs has previously been considered in [IP07, MN12]: unfortunately these protocols cannot be instantiated using OT-extension and therefore will not lead to practically efficient protocols (the security of these protocols is based on *strong* OT which, in a nutshell, requires the extra property that when executing several OTs in parallel, the receiver should not be able to correlate the answers with the queries beyond correlations which follow from the output).

[8] The key $k_\lambda^\star$ could not exists; e.g. if Y contains all the strings of λ bits.

[9] The only exception is the first OT execution where just two keys are using as input.

4.3 Definitions and tools

4.3.1 Two party computation

A two-party protocol problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality and denote it as $F = (F_1, F_2)$. That is, for every pair of inputs $x, y \in \{0,1\}^s$, the output-pair is a random variable $(F_1(x,y), F_2(x,y))$ ranging over pairs of strings. The first party (with input x) wishes to obtain $F_1(x,y)$ and the second party (with input y) wishes to obtain $F_2(x,y)$. We often denote such a functionality by $(x,y) \to (F_1(x,y), F_2(x,y))$.

Let $F = (F_1, F_2)$ be a probabilistic polynomial-time functionality and let $\Pi = (P_1, P_2)$ be a two-party protocol for computing F where P_1 and P_2 denote the two parties. The view of the party P_i ($i \in \{1,2\}$) during an execution of Π on (x,y) and security parameter s is denoted by $\mathsf{view}^{\Pi}_{P_i}(x, y, 1^s)$.

The output of the party P_i ($i \in \{1,2\}$) during an execution of Π on (x,y) and security parameter s is denoted by $\mathsf{output}^{\Pi}_{P_i}(1^s, x, y)$ and can be computed from its own view of the execution. We denote the joint output of both parties by $\mathsf{output}^{\Pi}(1^s, x, y) = (\mathsf{output}^{\Pi}_{P_1}(1^s, x, y), \mathsf{output}^{\Pi}_{P_2}(1^s, x, y))$.

Definition 14 (Secure two-party computation [HL10]). *Let $F = (F_1, F_2)$ be a functionality. We say that Π securely computes F in the presence of static semi-honest adversaries if there exist probabilistic polynomial-time algorithms Sim_{P_1} and Sim_{P_2} called simulators, such that*

$$\{(\mathsf{Sim}_{P_1}(1^s, x, F_1(x,y)), F(x,y))\}_{\{x,y,s\}} \approx \{\mathsf{view}^{\Pi}_{P_1}(1^s, x, y), \mathsf{output}^{\Pi}(1^s x, y)\}_{\{x,y,s\}}$$

and

$$\{(\mathsf{Sim}_{P_2}(1^s, y, F_2(x,y)), F(x,y))\}_{\{x,y,s\}} \approx \{\mathsf{view}^{\Pi}_{P_2}(1^s, x, y), \mathsf{output}^{\Pi}(1^s, x, y)\}_{\{x,y,s\}}$$

where $x, y \in \{0,1\}^\star$ such that $|x| = |y|$, and $s \in \mathbb{N}$.

4.3.2 *Special* private-key encryption

In our construction we use a private-key encryption scheme with two additional properties. The first is that given the key k, it is possible to efficiently verify if a given ciphertext is in the range of k. With the second property we require that an encryption under one key will fall in the range of an encryption under another key with negligible probability

As discussed in [LP09], it is easy to obtain a private-key encryption scheme with the properties that we require. According to Definition 2 of [LP09] we give the following definition.

Definition 15. *Let $\mathsf{Sym} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme and denote the range of a key in the scheme by $\mathsf{Range}_s(k) = \{\mathsf{Enc}(k,x)\}_{x \in \{0,1\}^s}$. Then*

1. *We say that* Sym *has an* efficiently verifiable range *if there exists a* PPT *algorithm M such that $M(1^s, k, c) = 1$ if and only if $c \in \mathsf{Range}_s(k)$. By convention, for every $c \notin \mathsf{Range}_s(k)$, we have that $\mathsf{Dec}(k,c) = \perp$.*

2. *We say that* Sym *has an* elusive range *if for every probabilistic polynomial-time machine $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that*

$$\mathrm{Prob}_{\mathrm{k} \leftarrow \mathsf{Gen}(1^{\mathrm{s}})}[\mathcal{A}(1^{\mathrm{s}}) \in \mathsf{Range}_{\mathrm{s}}(\mathrm{k})] < \nu(\mathrm{s}).$$

In order to make the security proof of our scheme easier, without loss of generality we assume Sym to be secure in the setting where the challenge messages $\mathbf{m}_0$ and $\mathbf{m}_1$ are lists of λ values. That is $\mathbf{m}_0 = \{m_0^1, \dots, m_0^\lambda\}$ and $\mathbf{m}_1 = \{m_1^1, \dots, m_1^\lambda\}$. The challenger, upon receiving these lists picks $b \leftarrow \{0,1\}$, defines an empty list $\mathbf{cx}$ and for $i = 1, \dots \lambda$ acts as follows:

1. computes $k_i \leftarrow \mathsf{Gen}(1^s)$;
2. computes $\mathsf{Enc}(k_i, m_b^i)$ and adds it to $\mathbf{cx}$.

The aim of the adversary is to guess the bit b having on input just $\mathbf{m}_0$, $\mathbf{m}_1$, $\mathbf{cx}$ and an auxiliary input z.

4.4 Our Protocol $\Pi^\in$

In this section we provide the formal description of our protocol $\Pi^\in = (\mathsf{S}, \mathsf{R})$ for the functionality $\mathcal{F}^\in = (\mathcal{F}^\in_{\mathsf{S}}, \mathcal{F}^\in_{\mathsf{R}})$ where

$$\mathcal{F}^\in_{\mathsf{S}} \colon \{\{0,1\}^\lambda\}^M \times \{\gamma^0, \gamma^1\} \times \{0,1\}^\lambda \longrightarrow \perp$$

and

$$\begin{aligned}\mathcal{F}^\in_{\mathsf{R}} \colon \{\{0,1\}^\lambda\}^M \times \{\gamma^0, \gamma^1\} \times \{0,1\}^\lambda &\longrightarrow \{\gamma^0, \gamma^1\} \\ (Y, x) &\longmapsto \begin{cases} \gamma^1 & \text{if } x \in Y \\ \gamma^0 & \text{otherwise} \end{cases}\end{aligned}$$

Where γ^0 and γ^1 are arbitrary strings and are part of the sender's input. Therefore our scheme protects both Y and γ^{1-b}, when γ^b is received by R.

For the formal description of $\Pi^\in$, we collapse the first and the second step from the information description in Section 4.2 into a single one. That is, instead of constructing the graph G, we only compute the keys at level i in order to feed the i-th OT execution with the correct inputs. The way in which the keys are computed is the same as the vertices for G are computed, we just do not need to physically construct G to allow S to efficiently compute the keys.

We use the following tools.

1. A protocol $\Pi_{\mathcal{OT}} = (\mathsf{S}_{\mathcal{OT}}, \mathsf{R}_{\mathcal{OT}})$ that securely (according to Definition 14) computes the following functionality

$$\begin{aligned}F_{\mathcal{OT}} \colon (\{0,1\}^\star \times \{0,1\}^\star) \times \{0,1\} &\longrightarrow \{\perp\} \times \{0,1\}^\star \\ ((c_0, c_1), b) &\longmapsto (\perp, c_b).\end{aligned}$$

2. A symmetric key encryption scheme $\mathsf{Sym} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with efficiently verifiable range and elusive range.

3. In our construction we make use of the following function

$$\begin{aligned}\delta \colon \mathbb{N} &\longrightarrow \mathbb{N} \\ i &\longmapsto \mathsf{min}\{2^i, |Y|\}.\end{aligned}$$

The function computes the maximum number of vertices that can appear in the level i of the graph G. As discussed before, the structure of G leaks information about Y. In order to avoid this information leakage about Y, it is sufficient to add some padding to the OT sender's input so that the input size become $|Y|$. Indeed, as observed above, every level contains at most $|Y|$ vertices. Actually, it is easy to see that $\mathsf{min}\{|Y|, 2^i\}$ represents a better upper bound on the number of vertices that the i-th level can contain given Y. Therefore, in order to compute the size of the padding for the sender's input we use the function δ.

4.4.1 Formal description

Common input: security parameter s and λ.
S*'s input:* a set Y of size M, $\gamma^0 \in \{0,1\}^s$ and $\gamma^1 \in \{0,1\}^s$.
R*'s input:* an element $x \in \{0,1\}^\lambda$.
First stage

1. For i=1, …, λ compute the sink key $k_i^\star \leftarrow \mathsf{Gen}(1^s)$.
2. S computes $k_0 \leftarrow \mathsf{Gen}(1^s), k_1 \leftarrow \mathsf{Gen}(1^s)$. For $b = 0,1$, if $b \notin \mathsf{Prefix}(Y,1)$ then set $k_b = k_1^\star$[10]. Set $(c_0^1, c_1^1) = (k_0, k_1)$.
3. S and R execute $\Pi_{\mathcal{OT}}$, where S acts as the sender $\mathsf{S}_{\mathcal{OT}}$ using (c_0^1, c_1^1) as input and R acts as a receiver using x_λ as input. When the execution of $\Pi_{\mathcal{OT}}$ ends R obtains $k_1 := c_{x_\lambda}^1$.

Second stage
For $i = 2, \dots, \lambda$:

1. S executes the following steps.
   - 1.1. Define the empty list c_0^i and for all $t \in \mathsf{Prefix}(Y, i-1)$ execute the following steps.

     If $t||0 \in \mathsf{Prefix}(Y,i)$ then compute $k_{t||0} \leftarrow \mathsf{Gen}(1^s)$ and add $\mathsf{Enc}(k_t, k_{t||0})$ to the list c_0^i. Otherwise, if $t||0 \notin \mathsf{Prefix}(Y,i)$ then compute and add $\mathsf{Enc}(k_t, k_i^\star)$ to the list c_0^i.
   - 1.2. If $|c_0^i| < \delta(i-1)$ then execute the following steps.
     - Compute and add $\mathsf{Enc}(k_{i-1}^\star, k_i^\star)$ to the list c_0^i.
     - For $i = 1, \dots, \delta(i-1) - |c_0^i|$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0)$ to c_0^i.[11]
   - 1.3. Permute the elements inside c_0^i.
   - 1.4. Define the empty list c_1^i and for all $t \in \mathsf{Prefix}(Y, i-1)$ execute the following step.

     If $t||1 \in \mathsf{Prefix}(Y,i)$ then compute $k_{t||1} \leftarrow \mathsf{Gen}(1^s)$ and add $\mathsf{Enc}(k_t, k_{t||1})$ to the list c_1^i. Otherwise, if $t||1 \notin \mathsf{Prefix}(Y,i)$ compute and add $\mathsf{Enc}(k_t, k_i^\star)$ to the list c_1^i.
   - 1.5. If $|c_1^i| < \delta(i-1)$ then execute the following steps.
     - Compute and add $\mathsf{Enc}(k_{i-1}^\star, k_i^\star)$ to the list c_1^i.
     - For $i = 1, \dots, \delta(i-1) - |c_1^i|$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0)$ to c_1^i.
   - 1.6. Permute the elements inside c_1^i.
2. S and R execute $\Pi_{\mathcal{OT}}$, where S acts as the sender $\mathsf{S}_{\mathcal{OT}}$ using (c_0^i, c_1^i) as input. and R acts as a receiver using $x_{\lambda-i+1}$ as input. When the execution of $\Pi_{\mathcal{OT}}$ ends, R obtains $c_{x_{\lambda-i+1}}^i$.

[10]We observe that if Y is not empty (like in our case) then there exists at most one bit b s.t. $b \notin \mathsf{Prefix}(Y,1)$.
[11]In this step, as well as in the step 1.5 of this stage, the function δ is used to compute the right amount of fake encryption to be added to the list that will we used as input of $\mathsf{R}_{\mathcal{OT}}$. The fake encryptions encrypts the value 0, but of course any other value could be used.

Third stage

1. S executes the following steps.

   1.1. Define the empty list l.

   1.2. For every $t \in \mathsf{Prefix}(Y, \lambda)$ compute and add $\mathsf{Enc}(k_t, \gamma^1)$ to l.

   1.3. If $|l| < 2^\lambda$ then compute and add $\mathsf{Enc}(k^\star_\lambda, \gamma^0)$ to l.

   1.4. Permute the elements inside l and send l to R.

2. R, upon receiving l execute the following steps.

   2.1. For $i = 2, \dots, \lambda$ execute the following step.

   For every element t in the list $c^i_{x_{\lambda-i+1}}$ compute $k \leftarrow \mathsf{Dec}(k_{i-1}, t)$. If $k \neq \perp$ then set $k_i = k$.

   2.2. For all $e \in l$ compute $\mathsf{out} \leftarrow \mathsf{Dec}(k_\lambda, e)$ and output out if and only if $\mathsf{out} \neq \perp$.

4.4.2 Complexity analysis

We focus our analysis on the described protocol without taking into account the many possible optimisations that we will describe in Sec. 4.5. In $\Pi^\in$, sender and receiver run λ executions of a 1-out-of-2 OT; in addition, they perform some symmetric key operations. More precisely, in order to compute the inputs for the i-th OT executions, with $i \in \{2, \dots, \lambda\}$, S computes $2 \cdot \mathsf{min}\{2^{i-1}, |Y|\}$ encryptions using the private-key encryption scheme Sym. We now observe that each encryption could contain a different key, and that this key needs to be generated by running $\mathsf{Gen}(\cdot)$.[12] This means that $4M$ represents an upper bound on the number of symmetric key operations performed by S in every OT execution. Moreover, in the last interaction with R, S computes M encryptions. Therefore, an upper bound on the number symmetric key operation performed by S is $(\lambda - 1) \cdot 4M + M + 2 \approx \lambda \cdot 4M$, where 2 represents the cost of running $\mathsf{Gen}(\cdot)$ twice in order to compute the two keys required to feed the first OT execution[13]. In every OT execution i, with $i \in \{2, \dots, \lambda\}$, R receives $\mathsf{min}\{2^{i+1}, |Y|\}$ encryptions, and tries to decrypt all of them. Moreover, in the last interaction with S, R receives M encryptions and tries to decrypt all of them as well. This means that the upper bound on the number of symmetric key operations made by R is $(\lambda - 1) \cdot M + M = \lambda \cdot M$. Following [PSZ16] we assume that 3 symmetric key operations are required for one OT execution. Therefore the total amount of symmetric key operations is $\lambda(4M + 3)$ for the sender and $\lambda(M + 3)$ for the receiver. In order to compare the efficiency of our protocol with the PSI protocols provided in [PSZ16] and to be consistent with their complexity analysis, we consider only the computation complexity for the party with the majority of the workload in the comparison.

Therefore we reach the conclusion that our protocol has better performance then all the circuit-based PSI approaches (which can be combined with further postprocessing) proposed in [PSZ16]. We note that, as described in Sec. 4.4 of [PSZ16], the approach based on evaluating the OPRF inside circuit is faster then any other PSI protocols if one set is much smaller the the other (like in the case of PSM), but in this case the output will necessarily leak to the receiver, which prevents composition with further 2PC protocol. We refer the reader to Table 7 of [PSZ16] for a detailed efficiency comparison between different PSI protocols). Finally, we observe that the complexities analysis proposed in [PSZ16] is related to PSI protocols, while in this section we have only compared the efficiency of the PSM subprotocol.

[12]We recall that $|Y| = M$ and that λ is the bit size of a set element.

[13]In this section, without loss of generality, we assume that λ is always greater than the security parameter s.

Communication complexity. The communication complexity of our protocol is dominated by the communication complexity of the underlying OT protocol $\Pi_{\mathcal{OT}} = (\mathsf{S}_{\mathcal{OT}}, \mathsf{R}_{\mathcal{OT}})$. Let $\mathsf{sOT}(D)$ be the amount of bit exchanged between $\mathsf{S}_{\mathcal{OT}}$ and $\mathsf{R}_{\mathcal{OT}}$ when $\mathsf{S}_{\mathcal{OT}}$ uses an input of size D, and let $\mathsf{sSYM}(A)$ be the size of a ciphertext for the encryption scheme Sym when a plaintex of size A is used. Then the communication complexity of our protocol is

$$\lambda \cdot \mathsf{sOT}(2 \cdot M \cdot \mathsf{sSYM}(\lambda)) + M \cdot \mathsf{sSYM}(\lambda)$$

where $2 \cdot M$ is the number of ciphertexts used as input of OT and M is the amount of ciphertexts that are sent in the last interaction between S and R. If we assume that a chipertext for Sym is roughly of size λ, and that $\Pi_{\mathcal{OT}}$ has a communication complexity that is approximately close to the size of the input used[14], we obtain that the overall communication complexity of our protocol is well approximated by $M\lambda \cdot (2\lambda + 1)$.

Round complexity: parallelizability of our scheme In the description of our protocol in Sec 4.4.1 we have the sender and the receiver engaging λ sequential OT executions. We now show that this is not necessary since the OT executions can be easily parallelized, given that each execution is independent from the other. That is, the output of a previous OT execution is not used in a later execution. For simplicity, we assume that $\Pi_{\mathcal{OT}}$ consists of just two rounds, where the first roun goes from the receiver to the sender, and the last goes in the opposite direction. We modify the description of the protocol of Sec 4.4.1 as follows.

- The step 3 of the *first stage* and the step 2 of the *second stage* are moved to the beginning of the *third stage*.
- When S sends the last round of $\Pi_{\mathcal{OT}}$, he also performs the step 1 of the *third stage*. Therefore the list l is sent together with the last rounds of the λ $\Pi_{\mathcal{OT}}$ executions.

Roughly, in this new protocol S first computes all the inputs $(k_0, k_1, c_0^1, c_1^1, \dots, c_0^\lambda, c_1^\lambda)$ for the OTs. Then, upon receiving λ first rounds of $\Pi_{\mathcal{OT}}$ computed by R using as input the bits of x, S sends λ second round of $\Pi_{\mathcal{OT}}$ together with the list l.

4.4.3 Security proof

Theorem 4. *Suppose $\Pi_{\mathcal{OT}}$ securely computes the 1-out-of-2 OT functionality $F_{\mathcal{OT}}$ and* Sym *is a symmetric key encryption scheme with efficiently verifiable range and elusive range, then $\Pi^{\in}$ securely computes the functionality $\mathcal{F}^{\in}$.*

Proof. In order to prove the security of $\Pi^{\in}$, according to Def. 14 we need show two probabilistic polynomial-time algorithms Sim_S and Sim_R called simulators, such that the following two conditions hold:

$$\begin{aligned}\{(\mathsf{Sim}_\mathsf{S}(1^s, Y, \gamma^0, \gamma^1, \mathcal{F}^{\in}_\mathsf{S}(Y, \gamma^0, \gamma^1, x)), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x))\}_{\{Y,x,s\}} \approx \\ \{\mathsf{view}^{\Pi^{\in}}_\mathsf{S}(1^s, Y, \gamma^0, \gamma^1, x), \mathsf{output}^{\Pi^{\in}}(1^s, Y, \gamma^0, \gamma^1, x)\}_{\{Y,x,s\}}\end{aligned} \tag{4.1}$$

$$\begin{aligned}\{(\mathsf{Sim}_\mathsf{R}(1^s, x, \mathcal{F}^{\in}_\mathsf{R}(Y, \gamma^0, \gamma^1, x)), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x))\}_{\{Y,x,s\}} \approx \\ \{\mathsf{view}^{\Pi^{\in}}_\mathsf{R}(1^s, Y, \gamma^0, \gamma^1, x), \mathsf{output}^{\Pi^{\in}}(1^s, Y, \gamma^0, \gamma^1, x)\}_{\{Y,x,s\}}\end{aligned} \tag{4.2}$$

where $Y \in \{\{0,1\}^\star\}^\star$, $x \in \{0,1\}^\star$, and $s \in \mathbb{N}$.

Therefore we divide our proof in two parts. In the former we show a PPT algorithm Sim_S that satisfies the property of the first point, and then a PPT algorithm Sim_R that satisfiy the requirment of the second point.

[14]This is actually true for the most common implementations of OT (OT extension).

Sim_S **description and proof of indistinguishability.** Sim_S runs S with some randomness r and the input Y. At this point Sim_S needs a strategy to acts as a receiver of OT in all the λ OT executions (without the receiver's input x). In order to do that, Sim_S runs the simulator of $\Pi_{\mathcal{OT}}$, that we call $\mathsf{Sim}_{\mathsf{S}_{\mathcal{OT}}}$ (and that exists by assumption), in every OT execution. We observe that in order to run $\mathsf{Sim}_{\mathsf{S}_{\mathcal{OT}}}$ in the i-th OT execution the inputs c_0^i and c_1^i need to be known. Clearly those values can be efficiently computed since the randomness r and the input Y used to run S are known.

We now show more formally how Sim_S works. Let $\mathsf{Sim}_{\mathsf{S}_{\mathcal{OT}}}$ be such that

$$\{\mathsf{Sim}_{\mathsf{S}_{\mathcal{OT}}}(1^s,(c_0,c_1),\perp),F_{\mathcal{OT}}((c_0,c_1),b)\}_{\{c_0,c_1,b,s\}}\approx$$
$$\{\mathsf{view}^{\Pi_{\mathcal{OT}}}_{\mathsf{S}_{\mathcal{OT}}}(1^s,(c_0,c_1),b),\mathsf{output}^{\Pi_{\mathcal{OT}}}(1^s,(c_0,c_1),b)\}_{\{c_0,c_1,b,s\}}$$

where $c_0,c_1\in\{0,1\}^\star$, $b\in\{0,1\}$, and $s\in\mathbb{N}$. Sim_S, on input Y and 1^s executes the following steps.

1. pick a $r\leftarrow\{0,1\}^s$ and runs S on input $1^s,Y$ using r as a randomness.
2. For every OT execution i, with $i=1,\dots,\lambda$, run $\mathsf{Sim}_{\mathsf{S}_{\mathcal{OT}}}$ on input 1^s, c_0^i and c_1^i, where c_0^i and c_1^i are computed using the same procedure that S uses.
3. Continue the execution against S as R would do.

In order to conclude this first part of the proof we just need to prove the following lemma.

Lemma 1.

$$\{(\mathsf{Sim}_\mathsf{S}(1^s,Y,\mathcal{F}^{\in}_\mathsf{S}(Y,\gamma^0,\gamma^1,x)),\mathcal{F}^{\in}(Y,\gamma^0,\gamma^1,x))\}\approx$$
$$\{\mathsf{view}^{\Pi^{\in}}_\mathsf{S}(1^s,Y,\gamma^0,\gamma^1,x),\mathsf{output}^{\Pi^{\in}}(1^s,Y,\gamma^0,\gamma^1,x)\}$$

where $Y\in\{\{0,1\}^\star\}^\star$, $x\in\{0,1\}^\star$, and $s\in\mathbb{N}$.[15]

Proof. The proof goes trough hybrid arguments starting from the real execution of $\Pi^{\in}$. We gradually modify the execution until the input of R is not needed anymore in such a way that the final hybrid represents the simulator Sim_S. We denote with $\mathsf{OUT}^{\mathcal{H}_i}_\mathsf{S}(1^s)$ the view of S in the hybrid experiment $\mathcal{H}_i$ with $i\in\{0,\dots,\lambda\}$. The hybrid experiments that we consider are the following.

1. $\mathcal{H}_0$ is identical to the real execution of $\Pi^{\in}$. More precisely $\mathcal{H}_0$ runs S using fresh randomness and interacts with him as R would do on input x.
2. $\mathcal{H}_i$ proceeds according to $\mathcal{H}_0$ with the difference that in the first i OT executions $\mathsf{Sim}_{\mathsf{S}_{\mathcal{OT}}}$ is used.

Since $\mathcal{F}^{\in}$ is a deterministic function we have that

$$\{\mathsf{view}^{\Pi^{\in}}_\mathsf{S}(1^s,Y,\gamma^0,\gamma^1,x),\mathcal{F}^{\in}(Y,\gamma^0,\gamma^1,x))\}\equiv$$
$$\{\mathsf{view}^{\Pi^{\in}}_\mathsf{S}(1^s,Y,\gamma^0,\gamma^1,x),\mathsf{output}^{\Pi^{\in}}(1^s,Y,\gamma^0,\gamma^1,x)\}\ .$$

Moreover we observe that

$$\{\mathsf{OUT}_{\mathcal{H}_0}(1^s),\mathcal{F}^{\in}(Y,\gamma^0,\gamma^1,x)\}=$$
$$\{\mathsf{view}^{\Pi^{\in}}_\mathsf{S}(1^s,Y,\gamma^0,\gamma^1,x),\mathcal{F}^{\in}(Y,\gamma^0,\gamma^1,x))\}$$

[15] To avoid overburdening the notation, here and in the rest of this chapter, we omit to specify the inputs domain when it is clear from the context.

and that

$$\{\mathsf{OUT}_{\mathcal{H}_\lambda}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} = \{(\mathsf{Sim}_{\mathsf{S}}(1^s, Y, \gamma^0, \gamma^1, \mathcal{F}^{\in}_{\mathsf{S}}(Y, \gamma^0, \gamma^1, x)), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x))\} \ .$$

Therefore the only thing that remains to argue is that

$$\{\mathsf{OUT}_{\mathcal{H}_{i-1}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} \approx \{\mathsf{OUT}_{\mathcal{H}_i}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$$

for $i = 1, \dots, \lambda$. We now show that if this statement does not hold then we can construct an adversary $\mathcal{A}^{\mathsf{S}_{\mathcal{OT}}}$ that breaks the security of $\Pi_{\mathcal{OT}}$ against malicious sender. Let $\mathsf{Sen}^{\mathsf{S}_{\mathcal{OT}}}$ be the challenger for the security game w.r.t. the security of $\Pi_{\mathcal{OT}}$ against malicious sender; the reduction works as follows.

1. $\mathcal{A}^{\mathsf{S}_{\mathcal{OT}}}$ runs S with randomness r and interacts with him according to $\mathcal{H}_{i-1}$ ($\mathcal{H}_i$) until the i-th OT execution.
2. At this point $\mathcal{A}^{\mathsf{S}_{\mathcal{OT}}}$ computes (c_0^i, c_1^i) and sends $((c_0^i, c_1^i), x_{\lambda-i+1})$ to $\mathsf{Sen}^{\mathsf{S}_{\mathcal{OT}}}$.
3. $\mathcal{A}^{\mathsf{S}_{\mathcal{OT}}}$ then acts as a proxy between $\mathsf{Sen}^{\mathsf{S}_{\mathcal{OT}}}$ and S.
4. When the interaction between $\mathsf{Sen}^{\mathsf{S}_{\mathcal{OT}}}$ and S is over, $\mathcal{A}$ continues the execution with S according to $\mathcal{H}_{i-1}$ ($\mathcal{H}_i$).

The security proof ends with the observation that if $\mathsf{Sen}^{\mathsf{S}_{\mathcal{OT}}}$ has used the simulator $\mathsf{Sim}_{\mathsf{S}_{\mathcal{OT}}}$ then the joint distribution of the view of S and $\mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)$ corresponds to $\{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}_i}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$, to $\{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}_{i-1}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$ otherwise.

□

□

$\mathsf{Sim}_{\mathsf{R}}$ description and proof of indistinguishability. At a very high level, $\mathsf{Sim}_{\mathsf{R}}$ runs R with some randomness r and the input x. $\mathsf{Sim}_{\mathsf{R}}$ then needs a strategy to acts as a sender of OT in all the λ OT executions (without sender's input Y). In order to do that, $\mathsf{Sim}_{\mathsf{R}}$ runs the simulator of $\Pi_{\mathcal{OT}}$, that we call $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ in every OT execution[16]. Moreover we need to feed the OT simulator with the correct input, depending on the value x. More precisely in the first OT execution $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ is run by using as input a key k_1. In the i-th OT execution (for $i = 2, \dots, \lambda$) the simulator will run using $x_{\lambda-i+1}$ and c^i. c^i contains encryptions of a fixed value, let say 0, computed using a fresh secret key (different for every ciphertext) and one encryption of the key k_i using the key k_{i-1}. After the λ OT executions $\mathsf{Sim}_{\mathsf{R}}$ sends to R M encryptions of 0 using a randomly generated secret key (also in this case a different secret key is used for each encryption of 0) and the encryption of the message $\mathsf{out} = \mathcal{F}^{\in}_{\mathsf{S}}(Y, \gamma^0, \gamma^1, x)$ using the key k_λ. We now show more formally how $\mathsf{Sim}_{\mathsf{R}}$ works. Let $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ be such that

$$\{\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}(1^s, b, c_b), F_{\mathcal{OT}}((c_0, c_1), b)\}_{\{c_0, c_1, b, s\}} \approx \{\mathsf{view}^{\Pi_{\mathcal{OT}}}_{\mathsf{R}_{\mathcal{OT}}}(1^s, (c_0, c_1), b), \mathsf{output}^{\Pi_{\mathcal{OT}}}(1^s, (c_0, c_1), b)\}_{\{c_0, c_1, b, s\}}$$

where $c_0, c_1 \in \{0,1\}^\star$, $b \in \{0,1\}$, and $s \in \mathbb{N}$.

$\mathsf{Sim}_{\mathsf{R}}$, on input x, out and 1^s executes the following steps.

[16]We recall that $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ exists by assumption.

1. Compute $\mathsf{k}_1 \leftarrow \mathsf{Gen}(1^s)$ ad run $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ on input $(1^s, x_\lambda, \mathsf{k}_1)$.
2. For $i = 2, \dots, \lambda$ execute the following steps.
   2.1. Define the empty list c^i. For $j = 1, \dots, \mathsf{min}\{2^i, |Y|\}-1$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0))$ to c^i.
   2.2. Compute $\mathsf{k}_i \leftarrow \mathsf{Gen}(1^s)$, and add $\mathsf{Enc}(\mathsf{k}_{i-1}, \mathsf{k}_i)$ to the list c^i.
   2.3. Permute the elements inside c^i.
   2.4. Run $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ on input $(1^s, x_{\lambda-i+1}, \mathsf{c}^i)$.
3. Define an empty list l.
4. For $i = 1, \dots M-1$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0)$ to l.
5. Add $\mathsf{Enc}(\mathsf{k}_\lambda, \mathsf{out})$ to l.
6. Permute the element inside l and send it.
7. Continue the execution according to R's description.

In order to conclude this latter part of the proof we need to prove the following lemma.

Lemma 2.

$$\{(\mathsf{Sim}_{\mathsf{R}}(1^s, x, \mathsf{out}, \mathcal{F}^{\in}_{\mathsf{R}}(Y, \gamma^0, \gamma^1, x)), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x))\}_{\{Y,x,s\}} \approx$$
$$\{\mathsf{view}^{\Pi^{\in}}_{\mathsf{R}}(1^s, Y, \gamma^0, \gamma^1, x), \mathsf{output}^{\Pi^{\in}}(1^s, Y, \gamma^0, \gamma^1, x)\}_{\{Y,x,s\}}$$

where $Y \in \{\{0,1\}^\star\}^\star$, $x \in \{0,1\}^\star$, *and* $s \in \mathbb{N}$.

Proof. The proof goes trough hybrid arguments starting from the real execution of $\Pi^{\in}$. We gradually modify the execution until the input of S (Y) is not needed anymore such that the final hybrid would represent the simulator $\mathsf{Sim}_{\mathsf{R}}$. We denote with $\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}_i}(1^s)$ the view of R in the hybrid experiment $\mathcal{H}_i$ with $i \in \{0, \dots, \lambda\}$.

1. $\mathcal{H}_0$ is identical to the real execution of $\Pi^{\in}$. More precisely $\mathcal{H}_0$ runs R using fresh randomness and interacts with him as S would do on input Y.
2. $\mathcal{H}_1$ proceeds according to $\mathcal{H}_0$ with the difference that in the first OT executions $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ is used on input $(1^s, x_\lambda, \mathsf{k}_1 \leftarrow \mathsf{Gen}(1^s))$.
3. $\mathcal{H}_i$ proceeds according to $\mathcal{H}_1$ with the difference that in the j-th OT executions, with $2 \leq j \leq i$, $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ is run on input $(1^s, x_{\lambda-j+1}, \mathsf{c}^j = c^j_{x_{\lambda-j+1}})$.
4. $\mathcal{H}^\star$ proceeds according to $\mathcal{H}_\lambda$ with the difference that in each OT execution i, with $2 \leq i \leq \lambda$, the input c^i for the simulator $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ is computed as follows.
   - For $j = 1, \dots, \mathsf{min}\{2^i, |Y|\} - 1$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0)$ to c^i.
   - Compute $\mathsf{k}_i \leftarrow \mathsf{Gen}(1^s)$, and add $\mathsf{Enc}(\mathsf{k}_{i-1}, \mathsf{k}_i)$ to the list c^i.
   - Permute the elements inside c^i.

   Moreover the first step of the third stage is performed as follows.

   - Define an empty list l.
   - For $i = 1, \dots M-1$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0)$ to l.

- Add $\mathsf{Enc}(\mathsf{k}_\lambda, \mathsf{out})$ to l.
- Permute the element inside l and send it to R.

Since $\mathcal{F}^{\in}$ is deterministic we have that

$$\{\mathsf{view}_{\mathsf{R}}^{\Pi^{\in}}(1^s, Y, \gamma^0, \gamma^1, x), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x))\} \equiv \\ \{\mathsf{view}_{\mathsf{R}}^{\Pi^{\in}}(1^s, Y, \gamma^0, \gamma^1, x), \mathsf{output}^{\Pi^{\in}}(1^s, Y, \gamma^0, \gamma^1, x)\} \ .$$

Moreover we observe that

$$\{\mathsf{OUT}_{\mathcal{H}_0}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} = \{\mathsf{view}_{\mathsf{R}}^{\Pi^{\in}}(1^s, Y, \gamma^0, \gamma^1, x), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x))\}$$

and that

$$\{\mathsf{OUT}_{\mathcal{H}^\star}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} = \{(\mathsf{Sim}_{\mathsf{R}}(1^s, x, \mathsf{out}, \mathcal{F}_{\mathsf{R}}^{\in}(Y, \gamma^0, \gamma^1, x)), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x))\} \ .$$

Therefore there are two things that remain to argue:

1. $\{\mathsf{OUT}_{\mathcal{H}_{i-1}}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} \approx \{\mathsf{OUT}_{\mathcal{H}_i}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$ for $i = 1, \ldots, \lambda$ and
2. $\{\mathsf{OUT}_{\mathcal{H}_\lambda}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} \approx \{\mathsf{OUT}_{\mathcal{H}^\star}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$.

We now start by showing that if the first statement does not hold for $i = 1$, then we can construct a adversary $\mathcal{A}^{\mathsf{S}_{\mathcal{OT}}}$ that breaks the security of $\Pi_{\mathcal{OT}}$ against malicious receiver. Let $\mathsf{Sen}^{\mathsf{R}_{\mathcal{OT}}}$ be the challenger for the security game w.r.t. the security of $\Pi_{\mathcal{OT}}$ against malicious receiver. The reduction works as follows.

1. $\mathcal{A}^{\mathsf{R}_{\mathcal{OT}}}$ runs R with randomness r, computes $k_0 \leftarrow \mathsf{Gen}(1^s)$, $k_1 \leftarrow \mathsf{Gen}(1^s)$ and sends $((k_0, k_1), x_\lambda)$ to $\mathsf{Sen}^{\mathsf{R}_{\mathcal{OT}}}$.
2. $\mathcal{A}^{\mathsf{R}_{\mathcal{OT}}}$ then acts as a proxy between $\mathsf{Sen}^{\mathsf{R}_{\mathcal{OT}}}$ and R.
3. When the interaction between $\mathsf{Sen}^{\mathsf{S}_{\mathcal{OT}}}$ and R is over, $\mathcal{A}^{\mathsf{R}_{\mathcal{OT}}}$ continues the execution with R according to $\mathcal{H}_0$ ($\mathcal{H}_1$).

This part of the security proof ends with the observation that if $\mathsf{Sen}^{\mathsf{R}_{\mathcal{OT}}}$ has used the simulator $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ then the joint distribution of the view of R and $\mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)$ corresponds to $\{\mathsf{OUT}_{\mathcal{H}_0}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$, to $\{\mathsf{OUT}_{\mathcal{H}_1}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$ otherwise.

The proof that

$$\{\mathsf{OUT}_{\mathcal{H}_{i-1}}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} \approx \{\mathsf{OUT}_{\mathcal{H}_i}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$$

for $i = 2, \ldots, \lambda$ follows the same arguments.

In order to prove that

$$\{\mathsf{OUT}_{\mathcal{H}_\lambda}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} \approx \{\mathsf{OUT}_{\mathcal{H}^\star}^{\mathsf{R}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} \ ,$$

thus concluding the lemma's security proof, we need to consider the following intermediate hybrid experiment $\mathcal{H}_y^\star$ with $y \in \{1, \ldots, \lambda\}$. The description of the hybrid experiment follows.

1. Compute $\mathsf{k}_1 \leftarrow \mathsf{Gen}(1^s)$ and run $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ on input $(1^s, x_\lambda, \mathsf{k}_1)$.
2. For $i = 2, \dots, y$ execute the following steps.
   - 2.1. Define the empty list c^i. For $j = 1, \dots, \mathsf{min}\{2^i, |Y|\}-1$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0))$ to c^i.
   - 2.2. Compute $\mathsf{k}_i \leftarrow \mathsf{Gen}(1^s)$, and add $\mathsf{Enc}(\mathsf{k}_{i-1}, \mathsf{k}_i)$ to the list c^i.
   - 2.3. Permute the elements inside c^i.
   - 2.4. Run $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ on input $(1^s, x_{\lambda-i+1}, \mathsf{c}^i)$.
3. For each $t \in \mathsf{Prefix}(Y, y) - \{x_\lambda \dots x_{\lambda-y+1}\}$ compute $k_t \leftarrow \mathsf{Gen}(1^s)$.
   If $x_\lambda \dots x_{\lambda-y+1} \in \mathsf{Prefix}(Y, y)$ then set $k_{x_\lambda \dots x_{\lambda-y+1}} = \mathsf{k}_y$, otherwise $k^\star_y = \mathsf{k}_y$.
4. For $i = y+1, \dots, \lambda$ execute the following steps.
   - 4.1. Define the empty list c^i and for each $t \in \mathsf{Prefix}(Y, i-1)$ execute the following steps.
     If $t||x_{\lambda-i+1} \in \mathsf{Prefix}(Y, i)$ then compute $k_{t||x_{\lambda-i+1}} \leftarrow \mathsf{Gen}(1^s)$ and add $\mathsf{Enc}(k_t, k_{t||x_{\lambda-i+1}})$ to the list c^i. Otherwise, if $t||x_{\lambda-i+1} \notin \mathsf{Prefix}(Y, i)$ then compute and add $\mathsf{Enc}(k_t, k^\star_i)$ to the list c^i.
   - 4.2. If $|\mathsf{c}^i| < \delta(i-1)$ then execute the following steps.
     - Compute and add $\mathsf{Enc}(k^\star_{i-1}, k^\star_i)$ to the list c^i.
     - For $i = 1, \dots, \delta(i-1) - |\mathsf{c}^i|$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0)$ to c^i.
   - 4.3. Permute the elements inside c^i.
   - 4.4. Run $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ on input $(1^s, x_{\lambda-i+1}, \mathsf{c}^i)$.
5. For every $t \in \mathsf{Prefix}(Y, \lambda)$ compute and add $\mathsf{Enc}(k_t, \gamma^1)$ to l.
6. If $|l| < 2^\lambda$ then compute and add $\mathsf{Enc}(k^\star_\lambda, \gamma^0)$ to l.
7. Permute the elements inside l and send l to R.

We now prove that

$$\{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}^\star_{y-1}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} \approx \{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}^\star_{y}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$$

for $y = 2, \dots, \lambda$. The proof proceeds by contradiction. Suppose that there exists some $y \in \{2, \dots, \lambda\}$ such that

$$\{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}^\star_{y-1}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} \approx \{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}^\star_{y}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$$

then we can construct ad adversary $\mathcal{A}^{\mathsf{Sym}}$ that breaks the security of the encryption scheme Sym. Let $\mathsf{Sen}^{\mathsf{Sym}}$ be the challenger for the security game w.r.t to Sym. Our adversary runs R with randomness r and executes the following steps.

1. Compute $\mathsf{k}_1 \leftarrow \mathsf{Gen}(1^s)$ and run $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ on input $(1^s, x_\lambda, \mathsf{k}_1)$.
2. For $i = 2, \dots, y-1$ execute the following steps.
   - 2.1. Define the empty list c^i. For $j = 1, \dots, \mathsf{min}\{2^i, |Y|\}-1$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0))$ to c^i.
   - 2.2. Compute $\mathsf{k}_i \leftarrow \mathsf{Gen}(1^s)$, and add $\mathsf{Enc}(\mathsf{k}_{i-1}, \mathsf{k}_i)$ to the list c^i.

2.3. Permute the elements inside c^i.

2.4. Run $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ on input $(1^s, x_{\lambda-i+1}, \mathsf{c}^i)$.

3. Define two empty lists $\mathbf{m}_0$ and $\mathbf{m}_1$ that will represent the challenge messages to be sent to $\mathsf{Sen}^{\mathsf{Sym}}$.

4. For each $t \in \mathsf{Prefix}(Y, y) - \{x_\lambda \dots x_{\lambda-y+1}\}$ compute $k_t \leftarrow \mathsf{Gen}(1^s)$ and add it to the list $\mathbf{m}_0$.

5. For $j = 1, \dots, |\mathsf{Prefix}(Y, y) - \{x_\lambda \dots x_{\lambda-y+1}\}|$ compute and add 0 to $\mathbf{m}_1$.

6. Send the challenge messages to $\mathsf{Sen}^{\mathsf{Sym}}$.

7. Upon receiving the challenge ciphertext $\mathbf{cx}$, set $\mathsf{c}^y = \mathbf{cx}$

8. For $j = 1, \dots, \mathsf{min}\{2^i, |Y|\} - |\mathbf{cx}| - 1$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0))$ to c^i.

9. Compute $\mathsf{k}_y \leftarrow \mathsf{Gen}(1^s)$, and add $\mathsf{Enc}(\mathsf{k}_{y-1}, \mathsf{k}_y)$ to the list c^i.

10. Permute the elements inside c^i.

11. Run $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ on input $(1^s, x_{\lambda-i+1}, \mathsf{c}^i)$.

12. If $x_\lambda \dots x_{\lambda-y+1} \in \mathsf{Prefix}(Y, y)$ then set $k_{x_\lambda \dots x_{\lambda-y+1}} = \mathsf{k}_y$, otherwise set $k^\star_y = \mathsf{k}_y$.

13. For $i = y + 1, \dots, \lambda$ execute the following steps.

    13.1. Define the empty list c^i and for each $t \in \mathsf{Prefix}(Y, i-1)$ execute the following steps.
    If $t||x_{\lambda-i+1} \in \mathsf{Prefix}(Y, i)$ then compute $k_{t||x_{\lambda-i+1}} \leftarrow \mathsf{Gen}(1^s)$ and add $\mathsf{Enc}(k_t, k_{t||x_{\lambda-i+1}})$ to the list c^i. Otherwise, if $t||x_{\lambda-i+1} \notin \mathsf{Prefix}(Y, i)$ then compute and add $\mathsf{Enc}(k_t, k^\star_i)$ to the list c^i.

    13.2. If $|\mathsf{c}^i| < \delta(i-1)$ then execute the following steps.
    - Compute and add $\mathsf{Enc}(k^\star_{i-1}, k^\star_i)$ to the list c^i.
    - For $i = 1, \dots, \delta(i-1) - |\mathsf{c}^i|$ compute and add $\mathsf{Enc}(\mathsf{Gen}(1^s), 0)$ to c^i.

    13.3. Permute the elements inside c^i.

    13.4. Run $\mathsf{Sim}_{\mathsf{R}_{\mathcal{OT}}}$ on input $(1^s, x_{\lambda-i+1}, \mathsf{c}^i)$.

14. For every $t \in \mathsf{Prefix}(Y, \lambda)$ compute and add $\mathsf{Enc}(k_t, \gamma^1)$ to l.

15. If $|l| < 2^\lambda$ then compute and add $\mathsf{Enc}(k^\star_\lambda, \gamma^0)$ to l.

16. Permute the elements inside l and send l to R.

This part of the security proof ends with the observation that if $\mathsf{Sen}^{\mathsf{Sym}}$ has used $\mathbf{m}_0$ then the joint distribution of the view of R and $\mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)$ corresponds to $\{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}^\star_{y-1}}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$, to $\{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}^\star_y}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$ otherwise.

Since the following two distributions coincide

$$\{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}_\lambda}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} = \{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}^\star_1}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$$

the to complete the entire security proof we just need to prove that $\{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}^\star_\lambda}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\} \approx \{\mathsf{OUT}^{\mathsf{R}}_{\mathcal{H}^\star}(1^s), \mathcal{F}^{\in}(Y, \gamma^0, \gamma^1, x)\}$. The indistinguishability between the two distributions can be proved by using arguments similar to the one used lately. That is, by proceedings by contradiction and constructing adversary that breaks the security of the encryption scheme Sym. □

4.5 Optimisations and extension

Point and Permute. In our protocol the receiver must decrypt every ciphertext at every layer to identify the correct one. This is suboptimal both because of the number of decryptions and because encryptions that have efficiently verifiable range necessarily have longer ciphertexts. This overhead can be removed using the standard *point-and-permute technique* [BMR90] which was introduced in the context of garbled circuits. Using this technique we can add to each key in each layer a *pointer* to the ciphertext in the next layer which can be decrypted using this key. This has no impact on security.

One-time Pad. It is possible to reduce the communication complexity of our protocol by using *one-time pad encryption* in the last $\log s$ layers of the graph, in the setting where the output values γ^0, γ^1 are such that $|\gamma^b| < s$. For instance, if the output values are bits (in case we combine our PSM with a GMW-style protocol), then the keys (and therefore the ciphertexts) used in the last layer of the graph only need to be 1 bit long. Unfortunately, since the keys in the second to last layer are used to mask up to two keys in the last layer, the keys in the second to last layer must be of length 2 and so on, which is why this optimization only gives benefits in the last $\log s$ layer of the graph.

PSM with Secret Shared Input. Our PSM protocol produces an output which can be post-processed using other 2PC protocol. It is natural to ask whether it is possible to design efficient PSM protocols that also work on encrypted or secret-shared inputs. We note here that our protocol can also be used in the setting in which the input string x is *bit-wise secret-shared* between the sender and the receiver i.e., the receiver knows a share r and the sender knows a share s s.t., $r \oplus s = x$. The protocol does not change for the receiver, who now inputs the bits of $r = r_\lambda, \ldots, r_1$ to the λ one-out-of-two OTs (instead of the bits of x as in the original protocol). The sender, at each layer i, will follow the protocol as described above if $s_i = 0$ and instead swap the inputs to the OT if $s_i = 1$. It can be easily verified that the protocol still produces the correct result and does not leak any extra information.

Keyword Search. Our PSM protocol outputs an encryption of a bit indicating whether $x \in Y$ or not. The protocol can be easily modified to output a value dependent on x itself and therefore implement "encrypted keyword search". That is, instead of having only two output strings γ^1, γ^0 representing membership and non-membership respectively, we can have $|Y|+1$ different output strings (one for each element $y \in Y$ and one for non-membership). This can be used for instance in the context where Y is a database containing id's y and corresponding values $v(y)$, and the output of the protocol should be an encryption of the value $v(x)$ if $x \in Y$ or a standard value $v(\perp)$ if $x \notin Y$. The modification is straightforward: instead of using all the keys in the last layer of the graph to encrypt the same value γ^1, use each key k_y to encrypt the corresponding value $v(y)$ and the sink key (which is use to encrypt γ^0 in our protocol) to encrypt the value $v(\perp)$.

PSI from PSM. We can follow the same approach of Phasing [PSSZ15, PSZ14] to turn our PSM protocol into a protocol for PSI. Given a receiver with input X and a sender with input Y the trivial way to construct PSI from PSM is to run $|X|$ copies of PSM, where in each execution the receiver inputs a different x from X and where the sender always inputs her entire set Y. As described above, the complexity of our protocol (as the complexity of Phasing) is proportional in the size of $|Y|$, so this naïve approach leads to quadratic complexity $O(|X| \cdot |Y|)$. Phasing deals with this using *hashing* i.e., by letting the sender and the receiver locally preprocess their inputs X, Y before engaging in the PSM protocols. The different hashing techniques are explained and analysed in [PSZ16][Section 3]. We present the intuitive idea and refer to their paper for details: in Phasing the receiver uses *Cuckoo hashing* to map X into a vector X' of size $\ell = O(|X|)$ such that all elements of X are present in X' and such that every $x'_i \in X'$ is either an element of X or a special $\perp$ symbol. The sender instead maps her set Y into $\ell = |X'|$ small buckets $Y'_1, \ldots, Y'_\ell$ such that every element $y \in Y$ is mapped into the "right bucket" i.e., the hashing has the property that if $y = x'_i$ for some i then y will end up in bucket Y'_i (and potentially in a few other buckets). Now Phasing uses the underlying PSM protocol to check whether x'_i is a member of Y'_i (for all i's), thus producing the desired result. The overall protocol complexity is now $O(|X'|\cdot|Y'_i|)$ which (by careful choice of the hashing parameters) can be made linear in input size i.e., the overall protocol has complexity $O(|X| + |Y|)$. Since this technique is agnostic of the underlying PSM protocol, we can apply the same technique to our PSM protocol to achieve a PSI protocol that produces encrypted output.

4.6 Applications

In this section we provide two examples of how our protocol can be used to implement more complex secure set operations. The examples show some guiding principles that can be used to design other applications based on our protocol.

As we will see, the major advantage provided by $\Pi^{\in}$ is that the output of the receiver can be an arbitrary value chosen by the sender as a function of x for each value $x \in Y \cup \{\perp\}$. This is in contrast with most of the approaches for set membership, where the value obtained by the receiver is a fixed value (e.g. 0) when $x \in Y$, or some random value otherwise.

Without loss of generality in the following applications only the receiver will learn the output of the computation. Moreover we assume that the size of X and Y is equal to the same value M.[17] Also for simplicity we will describe our application using the naïve PSI from PSM construction with quadratic complexity, but using the Phasing approach, as described in Sec. 4.5, it is possible to achieve linear complexity using hashing techniques. Finally, in both our applications we exploit the fact that additions can be performed locally (and for free) using secret-sharing based 2PC. In applications in which round complexity is critical, the protocols can be redesigned using garbled circuits computing the same functionality, since the garbled circuit can be sent from the sender to the other messages of the protocol. However in this case additions have to be performed inside the garbled circuit.

4.6.1 Computing statistics of the private intersection

Here we want to construct a protocol where sender and receiver have on input two sets, X and Y respectively, and want to compute some statistics on the intersections of their sets. For instance the receiver has a list of id's X and that the sender has a list of of id's Y and some corresponding values $v(Y)$ (thus we use the variant of our protocol for *keyword search* described in Section 4.5). At the end of the protocol the receiver should learn the average of $v(X \cap Y)$ (and not $|X \cap Y|$).

The main idea is the following: the sender and the receiver run M executions of our protocol where the receiver inputs a different x_i from X in each execution. The sender always inputs the same set Y, and chooses the $|Y| + 1$ outputs γ_i^y for all $y \in Y \cup \{\perp\}$ for all $i = 1, \ldots, M$ in the following way: γ_i^y is going to contain two parts, namely an arithmetic secret sharing of the bit indicating whether $x_i \in Y$ and an arithmetic secret sharing of the value $v(y)$. The arithmetic secret sharing will be performed using a modulo N large enough such that $N > M$ and $N > M \cdot V$ where V is some upper bound on $v(y)$ so to be sure that no modular reduction

[17]We assume this only to simplify the protocol description, indeed our protocol can be easily instantiated when the two sets have different size.

will happen when performing the addition of the resulting shares. Concretely the sender sets $\gamma_i^y = (-u_i^2 + 1 \mod N, -v_i^2 + v(y) \mod N)$ for all $y \in Y$ and $\gamma_i^\perp = (-u_i^2 \mod N, -v_i^2 \mod N)$. After the protocol the receiver defines her shares u_i^1, v_i^1 to the the shares contained in her output of the PSM protocol, and then both parties add their shares locally to obtain secret sharing of the size of the intersection and of the sum of the values i.e., $U^1 = \sum_i u_i^1$, $V^1 = \sum_i v_i^1$, $U^2 = \sum_i u_i^2$, and $V^2 = \sum_i v_i^2$. Now the parties check if (U^1, U^2) is a sharing of 0 and, if not, they compute and reveal the result of the computation $\frac{V^1+V^2}{U^1+U^2}$. Both these operations can be performed using efficient two-party protocols for comparison and division such as the one in [T+07, DNT12].

4.6.2 Threshold PSI

In this example we design a protocol $\Pi^t = (P_1^t, P_2^t)$ that securely compute the functionality $\mathcal{F}^t = (\mathcal{F}^t_{P_1^t}, \mathcal{F}^t_{P_2^t})$ where

$$\mathcal{F}^t_{P_1^t} \colon \{\{0,1\}^\lambda\}^M \times \{\{0,1\}^\lambda\}^M \longrightarrow \perp$$

and

$$\mathcal{F}^t_{P_2^t} \colon \{\{0,1\}^\lambda\}^M \times \{\{0,1\}^\lambda\}^M \longrightarrow \{\{0,1\}^\lambda\}^\star$$
$$(S_1, S_2) \longmapsto \begin{cases} S_1 \cap S_2 & \text{if } |S_1 \cap S_2| \geq t \\ \perp & \text{otherwise} \end{cases}$$

That is, the sender and the receiver have on input two sets, S_1 and S_2 respectively, and the receiver should only learn the intersection between these two sets if the size of the intersection is greater or equal than a fixed (public) threshold value t. In the case that the size of the intersection in smaller that t, then no information about S_1 is leaked to P_2^t and no information about S_2 is leaked to P_1^t. (This notion was recently considered in [HOS17] in the context of privacy-preserving ride-sharing).

As in the previous example, the sender and the receiver run M executions of our protocol where the receiver inputs a different x_i from S_2 in each execution. The sender always inputs the same set S_1, and chooses the two outputs γ_i^0, γ_i^1 in the following way: γ_i^b is going to contain two parts, namely an arithmetic secret sharing of 1 if $x_i \in Y$ or 0 otherwise, as well as encryption of the same bit using a key k. The arithmetic secret sharing will be performed using a modulo larger than M, so that the arithmetic secret sharings can be added to compute a secret-sharing of the value $|S_1 \cap S_2|$ with the guarantee that no overflow will occur. Then, the sender and the receiver engage in a secure-two party computation of a function that outputs the key k to the receiver if and only if $|S_1 \cap S_2| > t$. Therefore, if the intersection is larger than the threshold now the receiver can decrypt the ciphertext part of the γ values and learn which elements belong to the intersection. The required 2PC is a simple comparison with a known value (the threshold is public) which can be efficiently performed using protocols such as [GSV07, LT13].

Part II

Concurrent Non-Malleable Commitments

Chapter 5

Four-Round Concurrent Non-Malleable Commitments from One-Way Functions

5.1 Introduction

In this chapter we show how to construct a Four-Round Concurrent Non-Malleable Commitments from One-Way Functions. In order to construct such a commitment scheme, we provide a novel approach that allows to break the multiple-slot barrier for concurrent NM commitments, thus showing a 4-round scheme based on the sole existence of OWFs. While previous work relied on having either 1) stronger assumptions or 2) multiple rewind slots or 3) non-generic assumptions, in this work we introduce new techniques that allow to have just one rewind slot, minimal hardness assumptions and full concurrency.

We recall the contributions that will be provided in this chapter.

Non-malleable commitments w.r.t. non-aborting adversaries. We prove that a subprotocol of [GRRV14] is a 4-round statistically binding concurrent NM commitment scheme from OWFs (resp. a 3-round perfectly binding concurrent NM commitment scheme from 1-to-1 OWFs), if the adversary is restricted to playing well-formed commitments in the right sessions when receiving well formed commitments from the left sessions. We refer to this weaker security notion as concurrent weak non-malleability (wNM).

Simulation-Witness-Independence. We define a new security notion for argument systems w.r.t. man-in-the-middle attacks that we refer to as simulation-witness-independence (SimWI). This security notion seemingly is not implied by previous notions as simulation-extractability/soundness and strong non-malleable witness indistinguishability.

4-Round One-Many SimWI from OWFs. We then construct a 4-round one-many SimWI argument of knowledge for same specific languages by relying on OWFs only. This construction circumvents the major problem caused by the need of rewinding on the left to simulate and on the right to extract when there is only one available slot.

Concurrent wNM + One-Many SimWI ⇒ 4-Round Concurrent NM Commitments. We present our new paradigm consisting in combining the above two notions in a protocol that runs in parallel the concurrent wNM commitment scheme and the one-many SimWI argument of knowledge. Therefore as main result of this work we upgrade concurrent wNM to full-fledged concurrent non-malleability without any penalization in rounds and assumptions.

We now discuss in more details each of the above 4 contributions.

Weak Non-Malleable Commitments

We define commitment schemes enjoying a limited form of non-malleability[1].

Informally, we say that a commitment scheme is weak non-malleable (wNM) if it is non-malleable w.r.t. adversaries that never commit to $\perp$ when receiving honestly computed commitments. This form of non-malleability is significantly weaker than full-fledged non-malleability. Indeed, a full-fledged MiM $\mathcal{A}$ can for instance maul as follows: $\mathcal{A}$ creates a commitment of m_0 making use of messages computed by the sender in the left session so that if the sender commits to m_0 then the commitment of $\mathcal{A}$ is a well formed commitment of m_0, while instead if the sender commits to $m_1 \neq m_0$ then the commitment of $\mathcal{A}$ is not well formed and therefore corresponds to $\perp$. Such attacks can be explicitly instantiated as shown in [COSV16] where a generalization of the above $\mathcal{A}$ is used to prove that a preliminary version of the scheme of [GPR16] is not concurrent non-malleable.

While by itself the wNM guarantee is certainly unsatisfying as protection against MiM attacks, the design of a wNM commitment scheme can be an easier task and schemes with such light non-malleability flavor might exist with improved round complexity, efficiency and complexity assumptions compared to schemes achieving full-fledged non-malleability.

Concurrent wNM commitments. We show that a protocol due to [GRRV14] is a 4-round statistically binding concurrent wNM commitment scheme requiring OWFs only (resp., a 3-round perfectly binding concurrent wNM commitment scheme requiring 1-to-1 OWFs only). Moreover their protocol can be instantiated to be public coin. The security proof consists of some pretty straightforward observations on top of various useful lemmas already proven in [GRRV14]. Our contribution on wNM commitments therefore consists in 1) introducing and formalizing this notion; 2) observing the existence of a secure construction in previous work; 3) using it as one of the two main building blocks of our paradigm allowing to obtain 4-round concurrent (full-fledged) NM commitments from OWFs. A formal definition of weak NM commitments can be found in Sec. 5.2.1 (see Def. 20). The proof that a scheme proposed in [GRRV14] satisfies this notion can be found in Sec. 5.5.

Simulation-Witness-Independence.

We introduce a new security notion against MiM attacks to argument systems. We call our security notion *simulation-witness-independence* (SimWI) since it has similarities both with simulation extractability/soundness (see [PR05c, Sah99]) and with (strong) non-malleable witness indistinguishability [LPV09, OPV08] (sNMWI,NMWI). For simplicity we will discuss now the case of one prover and one verifier only, however our formal definition, construction and application will focus on the one-many case (i.e., up to 1 prover and polynomially many verifiers).

The 1st security flavor that our notion tries to capture is the concept that the view of a MiM in the real game should be simulatable. Therefore we will have an experiment corresponding to the real game where the MiM plays with a honest prover and a honest verifier, and an experiment corresponding to the simulated game that simply consists of the output of a stand-alone simulator that emulates the prover and runs the code of honest verifiers when interacting internally with the MiM[2].

While the above 1st security flavor guarantees that the statements proven by the MiM in the real-world experiment and in the simulated experiment are indistinguishable, there still is no

[1] We remark that Goyal in [Goy11] defined a weaker notion of non-malleable commitments (non-malleability w.r.t. replacement) that also had the goal to deal with commitments of $\perp$. While the goal is similar to our definition, the actual formulation is quite different.

[2] There is nothing surprising so far, this is just the concept of zero knowledge naturally augmented by extending the simulator with the behavior of honest verifiers to feed the MiM with messages belonging to the right sessions too.

guarantee that the MiM is unable to prove in the two experiments statements that are associated to witnesses belonging to distinguishable distributions. In other words, as 2nd flavor we want to capture the independence of the witnesses associated to the statements proven by the MiM with respect to the fact that the actual witness in the left session is used (this is the case of the real game) or is not used (this is the case of the simulated game). To avoid any ambiguity on which witness is associated to a statement, we associate to $\mathcal{NP}$ languages a non-negative integer γ. More precisely, for any $\mathcal{NP}$ language L we consider the largest non-negative integer γ such that for any $x \in L$ all witnesses of x have the same first γ bits[3]. The reason why we assign such a value γ to every $\mathcal{NP}$ language is that it fixes is some non-ambiguous way the input for the distinguisher of SimWI (indeed the input will be the first γ bits of any witness) and at same time the prefix of the witnesses of an instance can be recovered by extracting any witness.

The above 2nd flavor makes our security definition non-trivial. Indeed standard zero knowledge is clearly insufficient and the definition has strong connections with the (hard to achieve) concept of committed message in NM commitments[4]. One might think that some heavy machinery could already imply our new notion but instead, perhaps surprisingly, by taking into account all subtleties of the definitions it turns out that SimWI is not implied by simulation extractability, simulation soundness and sNMWI/NMWI. We stress that our goal is to get a *one-many 4*-round construction under minimal assumptions.

Comparison with simulation extractability and simulation soundness. Simulation extractability requires the simulator to output a transcript and witnesses for the statements appearing in the right sessions of the transcript.

Simulation soundness requires the MiM to fail in proving false statements when receiving simulated proofs of false statements.

SimWI requires the simulator to output a transcript that includes statements proven in right sessions. The distribution of the instance/witness pairs associated to those statements is required to be indistinguishable from the distribution of the instance/witness pairs associated to the statements proved by the MiM in the real game. Instead, in simulation extractability there is no requirement on the witness given in output by the simulator beyond being valid witnesses. Simulation soundness does not have any requirement on the witnesses associated to the statements proven by the MiM.

Comparison with sNMWI/NMWI. sNMWI considers two indistinguishable distributions of instance/witnesses pairs. Very informally, the requirement of sNMWI/NMWI is that the instance/witness pairs associated to the arguments given by the MiM in the right sessions be independent of the distribution from which the instance/witness pair of the argument given to the MiM in the left session has been sampled.

SimWI requires the existence of a simulator while instead sNMWI/NMWI only considers experiments where the actual prover plays.

One-Many SimWI From OWFs in 4 Rounds (i.e., in Just One Rewind Slot!). As discussed above, SimWI is an interesting security notions w.r.t. MiM attacks and similarly to all previous non-malleability notions is certainly non-trivial to achieve, especially when considering 1) the one-many case 2) only four rounds (i.e., one rewind slot) and 3) minimal assumptions. In this work we show how to construct a 4-round one-many SimWI argument of knowledge (AoK) from OWFs, therefore avoiding multiple rewind slots.

A common approach to construct 4-round zero-knowledge arguments (even without non-malleability requirements) relies on the FLS/FS paradigm [FLS90, FS90]. First there is a

[3]Note that when $\gamma = 0$ the 2nd security flavor is cancelled and SimWI becomes equivalent to zero knowledge. Furthermore when γ is equal to the largest witness size then the subclass of languages corresponds to $\mathcal{UP}$.

[4]We stress that the main goal of this work is to construct 4-round concurrent NM commitments from OWFs, and we will achieve it by making use of SimWI. As such, to avoid circularity, we can not use concurrent NM commitments to construct SimWI.

subprotocol useful to extract a trapdoor from the adversarial verifier. Then there is a witness-indistinguishable proof of knowledge (WIPoK) where the prover proves knowledge of either a witness for the statement or of the trapdoor. In order to save rounds the two subprotocols are parallelized.

The above common approach fails in presence of MiM attacks. The reason is that the MiM adversary can attack the witness indistinguishability (WI) of the WIPoK received in the left session in order to prove his statements (e.g., potentially false statement, statements with specific witnesses) in the right sessions. Using such a MiM to contradict the WI of the WIPoK is problematic since one should extract some useful information from the right session but this would require also to rewind the challenger of the WI of the WIPoK on the left.

We bypass the above difficulty as follows. Instead of relying on the WI of the WIPoK that requires two messages played by the challenger, we propose a construction where we essentially break the interactive challenger of WI into two non-interactive challengers. We implement this idea by relying on: 1) instance-dependent trapdoor commitments (IDTCom) and 2) special honest-verifier zero knowledge (special HVZK). More in details, let $(\pi_1, \pi_2, \pi_3, \pi_4)$ be the transcript of a delayed-input[5] 4-round special HVZK adaptive-input proof of knowledge (PoK). We require the prover to send an IDTCom $\mathtt{com}$ of π_2 that is opened, sending the opening $\mathtt{dec}$, only in the last round, when π_4 is sent. The actual transcript therefore becomes $(\pi_1, \mathtt{com}, \pi_3, (\pi_2, \mathtt{dec}, \pi_4))$.

Consider now an experiment where the trapdoor is known and π_2 can be opened arbitrarily. If the output of the experiment deviates from the original one, we will have a reduction to the trapdoorness of the IDTCom. The reduction is not problematic since the challenger of the trapdoorness is non-interactive, sending a pair (commitment, decommitment) that is either computed using the regular procedure or through the use of the trapdoor. Next, in another experiment we can replace the prover of the PoK with the special HVZK simulator that will compute π_2 and π_4 after having as input π_1 and π_3. Again, the output of this experiment will not deviate from the previous one otherwise we can show an adversary for the special HVZK property. The reduction again is not problematic since the challenger of special HVZK is non-interactive.

We implement the trapdoor-extraction subprotocol through OWFs by using as trapdoor knowledge of two signatures under the same public key sent by the verifier in the 1st round. The verifier will send a signature of one message (chosen by the prover) in the 3rd round. We will use a delayed-input special HVZK adaptive-input PoK where the prover proves knowledge of either a witness for the statement or of signatures of messages. The IDTCom will have the public key of the signature scheme as instance, therefore the simulator after having extracted the signatures will be able to equivocate the commitments. The security proof presents one more caveat. Once the simulator rewinds on the left to obtain the trapdoor it is not clear how to argue that the extraction from the right is meaningful since the extractor might simply obtain the same trapdoor and this is useless. More specifically, the adversary might be able to equivocate on the right, therefore the extractor of the PoK would fail, and the best we can get from such a binding violation is the trapdoor of the IDTCom played in the right session. This does not give any contradiction since the trapdoor of the right session had to be already known in order to answer twice (before and after the rewind) in the right session to the MiM. We resolve this problem by relying on a specific proof approach where while the initial transcript is generated by the simulator, when the extractions are played in the right sessions, the transcript of the left session is re-completed by running the prover of the special HVZK PoK. The reason why in this case the extraction on the right will succeed is that if we extract the trapdoor from the right session then this will also happen in the real game where the trapdoor is never used.

[5]By *delayed-input* we mean that the statement will be known only at the last round.

In turn it would break the security of the signature scheme.

Caveat: adaptive-input selection. We will give a formal definition that allows the MiM to select the instance/witness pair for the left session only at the end, while the MiM must fix the statement for a right session already when playing his first round in that session. Our construction satisfies this notion and even a more important form of adaptiveness. We allow the MiM to specify the statement in the last round of a right session, as long as the witness is already fixed when playing his first round in that session. The reason why we prove such more sophisticated form of adaptive-input selection is that it is required in our application for concurrent NM commitments. Ideally one would like to satisfy the best possible adaptive-input selection, in order to make this new primitive useful in a broader range of applications. However we can not prove our construction secure with fully adaptive-input selection since we are not able to extract the witness from a MiM selecting a new statement (with possibly a new witness) in the last round of a right session. Indeed we would end up having a certain statement in the transcript of the simulator and then a witness for another statement obtained through rewinds. This would negatively affect our proof approach. We leave as a very interesting open question the construction of a 4-round fully adaptive-input one-many SimWI AoK from OWFs.

4-Round Concurrent NM Commitments from OWFs.

We solve the problem left open by [GRRV14] by showing a 4-round concurrent NM commitment scheme relying on OWFs only. The new paradigm that we propose to obtain concurrent non-malleability consists in combining in parallel a 4-round public-coin concurrent wNM commitment scheme from OWFs Π_0, and a one-many 4-round SimWI argument of knowledge from OWFs Π_1.

The new paradigm. Π_0 is run in order to commit to the message m. Π_1 is instead used to prove knowledge of a valid message and randomness explaining the transcript of Π_0. The power of the new approach consists in using the above two tools that are in perfect synergy to defeat a concurrent MiM attack. The idea of the security proof is now quite simple. Since any one-many NM commitment is also many-many[6] NM, we focus the following discussion on the one-many case.

In the 1st experiment (the real game RG0) the sender commits to m_0. Clearly there can not be a commitment to $\perp$ on the right otherwise the soundness of Π_1 is contradicted. Symmetrically there is an experiment RG1 where the sender commits to m_1 and there is no commitment to $\perp$ on the right. Then we consider an hybrid game H0 where the simulator of one-many SimWI of Π_1 is used. Observe that if (by contradiction) the distribution of the messages committed on the right changes w.r.t. RG0 we have that also the distribution of the witnesses corresponding to the statements proved in Π_1 on the right changes. However this clearly violates SimWI. Therefore it must still be the case that a commitment played on the right corresponds to $\perp$ with negligible probability only. Symmetrically, there is an experiment H1 that is indistinguishable from RG1 and such that commitments played on the right are well formed (i.e., different from $\perp$). Therefore we can conclude that RG0 is indistinguishable from RG1 by noticing that H0 is indistinguishable from H1. Indeed, both H0 and H1 guarantee that the messages committed by the adversary on the right correspond to $\perp$ with negligible probability only. Summing up, a detectable deviation from H0 to H1 implies a contradiction of the concurrent wNM of Π_0[7]. This observation concludes the high-level overview of the security proof. However, some remarks are in order.

Remark 1: the required adaptive-input flavor. As specified in the previous section, our 4-round one-many SimWI AoK Π_1 is fully adaptive on the left but instead on the right

[6] A many-many NM commitment scheme can be also indicate as a concurrent NM commitment scheme. In the rest of the thesis we use the term concurrent.

[7] This reduction needs extra help, see Remark 2 below.

requires the witness to be fixed already in the first round of the MiM. The statements instead can be decided in the last round also in the right sessions. The flexibility with the statement is important since Π_0 is completed only in the last round and the entire transcript of Π_0 is part of the statement of Π_1. The lack of flexibility on the witness in the right sessions forces us to add one more requirement to Π_0. We need that message and randomness are already fixed in the 2nd round of Π_0, since they will be the witness for Π_1. This property is satisfied by the construction of [GRRV14] that we prove in Sec. 5.5 to be concurrent wNM.

Remark 2: on the need of public coins in Π_0. In a reduction we will have to simulate the last round of the receiver of Π_0 without knowing the randomness he used to compute the previous round. Obviously public coins are easy to simulate.

5.2 Definitions and tools

Definition 16 (One-way function (OWF))**.** *A function $f : \{0,1\}^\star \rightarrow \{0,1\}^\star$ is called one way if the following two conditions hold:*

- *there exists a deterministic polynomial-time algorithm that on input y in the domain of f outputs $f(y)$;*
- *for every* PPT *algorithm $\mathcal{A}$ there exists a negligible function ν, such that for every auxiliary input $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$:*

$$\text{Prob}\left[\, y \leftarrow \{0,1\}^\star : \mathcal{A}(f(y), z) \in f^{-1}(f(y)) \,\right] < \nu(\lambda).$$

We say that a OWF f is a 1-to-1 OWF *if $f(x) \neq f(y)$ $\forall (x,y) \in \{0,1\}^\star$.*
We say, also, that a OWF f is a one-way permutation (OWP) *if f is a permutation.*

Definition 17 (Following the notation of [CPS13])**.** *A triple of* PPT *algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathcal{V})$ *is called a* signature scheme *if it satisfies the following properties.*

Validity: *For every pair $(s,v) \leftarrow \mathsf{Gen}(1^\lambda)$, and every $m \in \{0,1\}^\lambda$, we have that*

$$\mathcal{V}(v, m, \mathsf{Sign}(s,m)) = 1.$$

Security: *For every* PPT *$\mathcal{A}$, there exists a negligible function ν, such that for all auxiliary input $z \in \{0,1\}^\star$ it holds that:*

$$\Pr[(s,v) \leftarrow \mathsf{Gen}(1^\lambda); (m,\sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}(s,\cdot)}(z,v) \wedge \mathcal{V}(v,m,\sigma) = 1 \wedge m \notin Q] < \nu(\lambda)$$

where Q denotes the set of messages whose signatures were requested by $\mathcal{A}$ to the oracle $\mathsf{Sign}(s,\cdot)$.

Definition 18 (Special Honest-Verifier Zero Knowledge (Special HVZK))**.** *Consider a public-coin proof/argument system $\Pi = (\mathcal{P}, \mathcal{V})$ for an $\mathcal{NP}$-language L where the verifier sends m messages of length $\ell_1, \ldots, \ell_m$. We say that Π is* Special HVZK if there exists a PPT simulator algorithm $\mathcal{S}$ that on input any $x \in L$, security parameter 1^λ and any $c_1 \in \{0,1\}^{\ell_1}, \ldots, c_m \in \{0,1\}^{\ell_m}$, outputs a transcript for proving $x \in L$ where $c_1, \ldots, c_m$ are the messages of the verifier, such that the distribution of the output of $\mathcal{S}$ is computationally indistinguishable from the distribution of a transcript obtained when $\mathcal{V}$ sends $c_1, \ldots, c_m$ as challenges and $\mathcal{P}$ runs on common input x and any w such that $(x,w) \in \mathsf{Rel_L}$.

In this chapter we consider the 3-round public-coin Special HVZK PoK proposed by Lapidot and Shamir [LS90], that we denote by LS. LS enjoys delayed-input completeness since the inputs for both $\mathcal{P}$ and $\mathcal{V}$ are needed only to play the last round, and only the length of the instance is needed earlier. LS also enjoys adaptive-input PoK. In particular in our work we use use a 4-round delayed-input, special HVZK, adaptive-input AoK, that is a variant of LS [Fei90] that relies on OWFs only. The additional round is indeed needed to instantiate the commitment scheme used in LS under any OWF.

5.2.1 Non-Malleable Commitments

Here we follow [LPV08]. Let $\Pi = (\mathsf{Sen}, \mathsf{Rec})$ be a statistically binding commitment scheme and let λ be the security parameter. Consider MiM adversaries that are participating in left and right sessions in which $\mathsf{poly}(\lambda)$ commitments take place. We compare between a MiM and a simulated execution. In the MiM execution the adversary $\mathcal{A}$, with auxiliary information z, is simultaneously participating in $\mathsf{poly}(\lambda)$ left and right sessions. In the left sessions the MiM adversary $\mathcal{A}$ interacts with $\mathsf{Sen}_1, \dots, \mathsf{Sen}_{\mathsf{poly}(\lambda)}$ receiving commitments to values $m_1, \dots, m_{\mathsf{poly}(\lambda)}$ using identities $\mathtt{id}_1, \dots, \mathtt{id}_{\mathsf{poly}(\lambda)}$ of its choice. In the right session $\mathcal{A}$ interacts with $\mathsf{Rec}_1, \dots, \mathsf{Rec}_{\mathsf{poly}(\lambda)}$ attempting to commit to a sequence of related values $\tilde{m}_1, \dots, \tilde{m}_{\mathsf{poly}(\lambda)}$ again using identities of its choice $\tilde{\mathtt{id}}_1, \dots, \tilde{\mathtt{id}}_{\mathsf{poly}(\lambda)}$. If any of the right commitments is invalid, or undefined, its value is set to $\perp$. For any i such that $\tilde{\mathtt{id}}_i = \mathtt{id}_j$ for some j, set $\tilde{m}_i = \perp$ (i.e., any commitment where the adversary uses the same identity of one of the honest senders is considered invalid). Let $\mathsf{mim}_{\Pi}^{\mathcal{A}, m_1, \dots, m_{\mathsf{poly}(\lambda)}}(z)$ denote a random variable that describes the values $\tilde{m}_1, \dots, \tilde{m}_{\mathsf{poly}(\lambda)}$ and the view of $\mathcal{A}$, in the above experiment. In the simulated execution, an efficient simulator S directly interacts with $\mathsf{Rec}_1, \dots, \mathsf{Rec}_{\mathsf{poly}(\lambda)}$. Let $\mathsf{sim}_{\Pi}^{S}(1^\lambda, z)$ denote the random variable describing the values $\tilde{m}_1, \dots, \tilde{m}_{\mathsf{poly}(\lambda)}$ committed by S, and the output view of S; whenever the view contains in the i-th right session the same identity of any of the identities of the left sessions, then m_i is set to $\perp$.

In all the chapter we denote by $\tilde{\delta}$ a value associated with the right session (where the adversary $\mathcal{A}$ plays with a receiver) where δ is the corresponding value in the left session. For example, the sender commits to v in the left session while $\mathcal{A}$ commits to $\tilde{v}$ in the right session.

Definition 19 (Concurrent NM commitment scheme [LPV08]). *A commitment scheme is* concurrent NM with respect to commitment *(or a many-many NM commitment scheme) if, for every* PPT *concurrent MiM adversary $\mathcal{A}$, there exists a* PPT *simulator S such that for all $m_i \in \{0,1\}^{\mathsf{poly}(\lambda)}$ for $i = 1, \dots, \mathsf{poly}(\lambda)$ the following ensembles are computationally indistinguishable:*
$\{\mathsf{mim}_{\Pi}^{\mathcal{A}, m_1, \dots, m_{\mathsf{poly}(\lambda)}}(z)\}_{z \in \{0,1\}^\star} \approx \{\mathsf{sim}_{\Pi}^{S}(1^\lambda, z)\}_{z \in \{0,1\}^\star}$.

As in [LPV08] we also consider relaxed notions of concurrent non-malleability: one-many and one-one NM commitment schemes. In a one-many NM commitment scheme, $\mathcal{A}$ participates in one left and polynomially many right sessions. In a one-one (i.e., a stand-alone secure) NM commitment scheme, we consider only adversaries $\mathcal{A}$ that participate in one left and one right session. We will make use of the following proposition of [LPV08].

Proposition 1. *Let* $(\mathsf{Sen}, \mathsf{Rec})$ *be a one-many NM commitment scheme. Then,* $(\mathsf{Sen}, \mathsf{Rec})$ *is also a concurrent (i.e., many-many) NM commitment scheme.*

We say that a commitment is valid or well formed if it can be decommitted to a message $m \neq \perp$. Following [LP11] we say that a MiM is *synchronous* if it "aligns" the left and the right sessions; that is, whenever it receives message i on the left, it directly sends message i on the right, and vice versa.

5.2.2 New Definitions: weak NM and SimWI

Definition 20 (*weak* NM commitment scheme)**.** *A commitment scheme is* weak *one-one (resp., one-many) non-malleable if it is a one-one (resp., one-many) NM commitment scheme with respect to MiM adversary that when receiving a well formed commitment in the left session, except with negligible probability computes well formed commitments (i.e., the computed commitments can be opened to messages $\neq \perp$) in the right sessions.*

In the rest of the thesis, following [GRRV14], we assume that identities are known before the protocol begins, though strictly speaking this is not necessary, as the identities do not appear in the protocol until after the first committer message. The MiM can choose his identity adversarially as long as it differs from the identities used by honest senders. As already observed in previous work, when the identity is selected by the sender the id-based definitions guarantee non-malleability as long as the MiM does not behave like a proxy (an unavoidable attack). Indeed the sender can pick as `id` the public key of a signature scheme signing the transcript. The MiM will have to use a different `id` or to break the signature scheme.

Simulation-witness-independence (SimWI) for L^γ. We define SimWI for an $\mathcal{NP}$ language L associating to the language a non-negative integer γ. Roughly speaking all witnesses of an instance have in common the first γ bits, and this property holds for all instances of L. More formally we will consider γ as a non-negative integer such that for any $x \in L$ it holds that any witness w of x can be parsed as $w = \alpha|\beta$, where $|\alpha| = \gamma$, and α is the same for all witnesses of x. In order to easy the notation, we will note denote by L^γ the $\mathcal{NP}$ language having the above prefix γ. We will say that L^γ is γ-prefix language meaning that for any instance x of L^γ all witnesses of x have the same first γ bits.

When defining SimWI we will consider the one-many case since this is what we will use in the next part of this chapter. Adapting the definition to the one-one case and to the fully concurrent case is straight-forward.

Discussion on adaptive-input selection and black-box simulation. Since our definition considers a real game where the MiM plays with at most one prover and polynomially many verifiers, and a simulated game that consists of an execution of a stand-alone simulator, a natural definition would require the indistinguishability of the two games for any $x \in L^\gamma$, giving to the prover as input also a witness. This definition however would be difficult to use when the argument of knowledge is played as a subprotocol of a larger protocol, especially if it is played in parallel with other subprotocols and the adversary contributes in selecting the statement for the left session. More specifically applications require a security definition that features a delayed-input property so that players start the protocol with the common input that is still undefined, and that will be defined later potentially with the contribution of the adversary. Therefore in our definition we will allow the adversary to explicitly select the statement, and as such the adversary will provide also the witness for the prover. The simulated game however will filter out the witness so that the simulator will receive only the instance. This approach strictly follows the one of [SCO+01] where adaptive-input selection is explicitly allowed and managed in a similar way. As final remark, our definition will require the existence of a black-box simulator since a non-black-box simulator could retrieve from the code of the adversary the witness for the adaptively generated statement. The non-black-box simulator could then run the honest prover procedure, therefore canceling completely the security flavor of the simulation paradigm.

Definition. Let $\Pi = (\mathcal{P}, \mathcal{V})$ be an argument system for an γ-prefix language L^γ and let $\mathsf{Rel}_{\mathsf{L}^\gamma}$ be the corresponding witness relation. Consider a PPT MiM adversary $\mathcal{A}$ that is simultaneously participating in one left session and $\mathsf{poly}(\lambda)$ right sessions. When the execution starts, all parties receive as a common input the security parameter 1^λ then $\mathcal{A}$ chooses the statement $x \in L^\gamma$ and witness w s.t. $(x, w) \in \mathsf{Rel}_{\mathsf{L}^\gamma}$ and sends them to $\mathcal{P}$, furthermore $\mathcal{A}$ receives as auxiliary input $z \in \{0,1\}^\star$.

In the left session an honest prover $\mathcal{P}$ interacting with $\mathcal{A}$ proves the membership of x in L^γ. In the $\mathsf{poly}(\lambda)$ right sessions, $\mathcal{A}$ proves the membership in L^γ of instances $\tilde{x}_1, \dots, \tilde{x}_{\mathsf{poly}(\lambda)}$ of his choice to the honest verifiers $\mathcal{V}_1, \dots, \mathcal{V}_{\mathsf{poly}(\lambda)}$. For simplicity, in this definition we consider an adversary $\mathcal{A}$ that chooses the statement to be proved in the 1st round that he plays in every right sessions[8].

Let $\{\mathsf{wimim}_\Pi(1^\lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^\star}$ be a random variable that describes the following 3 values: 1) the view of $\mathcal{A}$ in the above experiment, 2) the output of $\mathcal{V}_i$ for $i = 1, \dots, \mathsf{poly}(\lambda)$ and 3) the first γ-bits $\tilde{w}_1^\gamma, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}^\gamma$ of the corresponding witnesses $\tilde{w}_1, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}$ w.r.t. the instances $\tilde{x}_1, \dots, \tilde{x}_{\mathsf{poly}(\lambda)}$ that are part of $\mathcal{A}$'s view except that $\tilde{w}_i^\gamma = \bot$ if $\mathcal{V}_i$ did not output 1, with $i = 1, \dots, \mathsf{poly}(\lambda)$.

Let $\{\mathsf{sim}_\Pi^\mathcal{S}(1^\lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^\star}$ be a random variable that describes the following 3 values: 1) and 2) correspond to the output of $\mathcal{S}$, 3) consists of the first γ-bits $\tilde{w}_1^\gamma, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}^\gamma$ of the corresponding witnesses $\tilde{w}_1, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}$ w.r.t. the instances $\tilde{x}_1, \dots, \tilde{x}_{\mathsf{poly}(\lambda)}$ that appear in the MiM view of the output of $\mathcal{S}$ except that $w_i^\gamma = \bot$ if $b_i = 0$. The output of $\mathcal{S}$ is composed by the following two values: 1) a MiM view and 2) bits $b_1, \dots, b_{\mathsf{poly}(\lambda)}$.

$\mathcal{S}$ has black-box access to $\mathcal{A}$ and has the goal to emulate the prover without having a witness, while perfectly emulating the verifiers of the right sessions. Therefore $\mathcal{S}$ rewinds only when playing as prover[9] and every instance/witness pair (x, w) given in output by $\mathcal{A}$ is replaced by $(x, \bot)$ and then returned to $\mathcal{S}$ (i.e., the simulator runs without the witness w for the instance x chosen by $\mathcal{A}$).

Definition 21 (SimWI). *An argument system $\Pi = (\mathcal{P}, \mathcal{V})$ for an γ-prefix language L^γ with witness relation $\mathsf{Rel}_{\mathsf{L}^\gamma}$ is* SimWI *if there exists an expected polynomial-time simulator $\mathcal{S}$ such that for every MiM adversary $\mathcal{A}$ that participates in one left session and $\mathsf{poly}(\lambda)$ right sessions the ensembles $\{\mathsf{wimim}_\Pi(1^\lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^\star}$ and $\{\mathsf{sim}_\Pi^\mathcal{S}(1^\lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^\star}$ are computationally indistinguishable over λ.*

5.3 4-Round One-Many SimWI From OWFs

We now show our construction of a 4-round argument of knowledge $\mathsf{SWI} = (\mathcal{P}^{\mathsf{swi}}, \mathcal{V}^{\mathsf{swi}})$ for the γ-prefix language L^γ that is one-many SimWI and can be instantiated using any OWF. We will need the following tools:

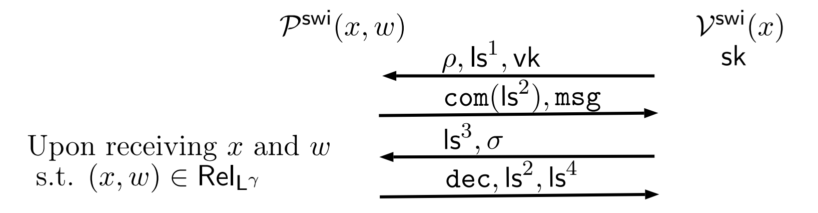
1. a signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathcal{V})$;
2. a 2-round IDTC scheme $\mathsf{TC}_\Sigma = (\mathsf{Sen}_\Sigma, \mathsf{Rec}_\Sigma, \mathsf{TFake}_\Sigma)$ for the following $\mathcal{NP}$-language

$$L_\Sigma = \big\{\mathsf{vk} : \exists\ (\mathtt{msg}_1, \mathtt{msg}_2, \sigma_1, \sigma_2) \text{ s.t. } \mathcal{V}(\mathsf{vk}, \mathtt{msg}_1, \sigma_1) = 1$$
$$\mathtt{AND}\ \mathcal{V}(\mathsf{vk}, \mathtt{msg}_2, \sigma_2) = 1\ \mathtt{AND}\ \mathtt{msg}_1 \neq \mathtt{msg}_2\big\};$$

[8]Our construction will satisfy a much stronger notion where in the left session $\mathcal{A}$ can choose statement and witness in the last round, while in the right sessions $\mathcal{A}$ can choose the statement in the very last round, as long as the witness is already fixed in the second round.

[9]The motivation behind this definitional choice is that $\mathcal{S}$ is supposed to be not more than an extended zero-knowledge simulator that takes care also of the honest behavior of the verifiers since $\mathcal{A}$ expects to play with them. Instead allowing $\mathcal{S}$ to have any behavior on the right would hurt the power of SimWI in composing in parallel with other protocols. Indeed if $\mathcal{S}$ rewinds on the right as verifier, it would in turn rewind also the left player of the external protocol that is played in parallel. This would hurt the security of the overall scheme whenever the external protocol is not resettably secure. We will indeed compose a SimWI AoK with a weak NM commitment scheme that is not resettably secure.

3. a 4-round delayed-input public-coin Special HVZK (Def. 18) proof system $\mathsf{LS} = (\mathcal{P}, \mathcal{V})$ for the γ-prefix language L^γ that is adaptive-input PoK for the corresponding relation $\mathsf{Rel}_{\mathsf{L}^\gamma}$.



$\mathcal{P}^{\mathsf{swi}}(x, w)$
$\mathcal{V}^{\mathsf{swi}}(x)$
sk
$\rho, \mathsf{ls}^1, \mathsf{vk}$
$\mathtt{com}(\mathsf{ls}^2), \mathtt{msg}$
Upon receiving x and w s.t. $(x, w) \in \mathsf{Rel}_{\mathsf{L}^\gamma}$
ls^3, σ
$\mathtt{dec}, \mathsf{ls}^2, \mathsf{ls}^4$

– $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(\lambda)$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathtt{msg})$;
– $\mathtt{com}(\mathsf{ls}^2)$ is the commitment computed by running Sen_Σ for the language L_Σ and instance vk on input ρ (i.e., the 1st round of the commitment scheme), in order to commit to ls^2 ($\mathtt{dec}$ represents the decommitment information of $\mathtt{com}(\mathsf{ls}^2)$);
– $(\mathsf{ls}^1, \mathsf{ls}^2, \mathsf{ls}^3, \mathsf{ls}^4)$ is the transcript of LS for proving that $x \in L^\gamma$.

Figure 5.1: Our SimWI AoK SWI from OWFs.

Let $x \in L^\gamma$ be the statement that $\mathcal{P}^{\mathsf{swi}}$ wants to prove, and w a witness s.t. $(x, w) \in \mathsf{Rel}_{\mathsf{L}^\gamma}$. The high-level idea of our protocol is depicted in Fig. 5.1. In the 1st round the verifier $\mathcal{V}^{\mathsf{swi}}$ computes and sends the 1st round ls^1 of LS, computes a pair of signature and verification keys $(\mathsf{sk}, \mathsf{vk})$, sends the verification key vk to $\mathcal{P}^{\mathsf{swi}}$ and computes and sends the 1st round ρ of TC_Σ by running Rec_Σ on input 1^λ and the instance $\mathsf{vk} \in L_\Sigma$. Then $\mathcal{P}^{\mathsf{swi}}$ on input x, w and the received 1st round, computes the 2nd round ls^2 of LS and runs Sen_Σ on input 1^λ, vk, ρ and message ls^2 thus obtaining a pair $(\mathtt{com}, \mathtt{dec})$. $\mathcal{P}^{\mathsf{swi}}$ sends $\mathtt{com}$ and a random message $\mathtt{msg}$ to $\mathcal{V}^{\mathsf{swi}}$. In the 3rd round $\mathcal{V}^{\mathsf{swi}}$ sends the 3rd round ls^3 of LS and a signature σ (computed using sk) of the message $\mathtt{msg}$. In the last round $\mathcal{P}^{\mathsf{swi}}$ verifies whether or not σ is a valid signature for $\mathtt{msg}$. If σ is a valid signature, then $\mathcal{P}^{\mathsf{swi}}$, using x, w and ls^3, computes the 4th round ls^4 of LS and sends $\mathtt{dec}$, ls^2 and ls^4 to $\mathcal{V}^{\mathsf{swi}}$. At this point $\mathcal{V}^{\mathsf{swi}}$ outputs 1 iff Rec_Σ on input $\mathsf{vk}, \mathtt{com}, \mathtt{dec}, \mathsf{ls}^2$ accepts $(\mathsf{ls}^2, \mathtt{dec})$ as a decommitment of $\mathtt{com}$ and the transcript for LS is accepting for $\mathcal{V}$ with respect to the instance x. We remark that to execute LS the instance is not needed until the last round but the instance length is required from the onset of the protocol.

The Fig. 5.2 describes in details our SimWI AoK SWI.

Theorem 5. *Assuming OWFs,* $\mathsf{SWI} = (\mathcal{P}^{\mathsf{swi}}, \mathcal{V}^{\mathsf{swi}})$ *is a 4-round one-many SimWI AoK for γ-prefix language.*

We divide the security proof in three parts, proving that SWI enjoys delayed-input completeness, adaptive-input AoK and SimWI. Before that, we recall that LS can be constructed from OWFs (see Sec. 5) as well as Σ using [Rom90]. We also observe that if Σ relies on OWFs, then also TC_Σ can be constructed from OWFs (see Sec. 5).

Delayed-Input Completeness. The completeness follows directly from the completeness of LS, the correctness of TC_Σ and the validity of Σ. We observe that, due to the delayed-input property of LS, the statement x (and the respective witness w) are used by $\mathcal{P}^{\mathsf{swi}}$ only to compute the last round. Therefore SWI enjoys delayed-input completeness.

Adaptive-Input Argument of Knowledge. In order to prove that SWI enjoys adaptive-input AoK for $\mathsf{Rel}_{\mathsf{L}^\gamma}$, we need to show an efficient extractor E that outputs the witnesses for the statements proved by an adversarial prover $\mathcal{P}^{\mathsf{swi}\star}$. E simply runs ExtLS, the adaptive-input PoK extractor of LS, in every right session, and outputs what ExtLS outputs. More precisely E

Common input: security parameter λ, instance $x \in L^{\gamma}$, instance length ℓ.
Input to $\mathcal{P}^{\mathsf{swi}}$: w s.t. $(x, w) \in \mathsf{Rel}_{\mathsf{L}^{\gamma}}$, with x, w available only in the 4th round.
Commitment phase:

1. $\mathcal{V}^{\mathsf{swi}} \rightarrow \mathcal{P}^{\mathsf{swi}}$
    1. Run $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^{\lambda})$.
    2. Run $\mathcal{V}$ on input 1^{λ} and ℓ thus obtaining the 1st round ls^1 of LS.
    3. Run Rec_{Σ} on input 1^{λ} and vk thus obtaining ρ.
    4. Send $(\mathsf{vk}, \mathsf{ls}^1, \rho)$ to $\mathcal{P}^{\mathsf{swi}}$.
2. $\mathcal{P}^{\mathsf{swi}} \rightarrow \mathcal{V}^{\mathsf{swi}}$
    1. Run $\mathcal{P}$ on input 1^{λ}, ℓ and ls^1 thus obtaining the 2nd round ls^2 of LS.
    2. Run Sen_{Σ} on input 1^{λ}, vk, ρ and message ls^2 to compute the pair $(\mathtt{com}, \mathtt{dec})$.
    3. Pick a message $\mathtt{msg} \leftarrow \{0,1\}^{\lambda}$.
    4. Send $(\mathtt{com}, \mathtt{msg})$ to $\mathcal{V}^{\mathsf{swi}}$.
3. $\mathcal{V}^{\mathsf{swi}} \rightarrow \mathcal{P}^{\mathsf{swi}}$
    1. Run $\mathcal{V}$ thus obtaining the 3rd round ls^3 of LS.
    2. Run $\mathsf{Sign}(\mathsf{sk}, \mathtt{msg})$ thus obtaining a signature σ of the message $\mathtt{msg}$.
    3. Send (ls^3, σ) to $\mathcal{P}^{\mathsf{swi}}$.
4. $\mathcal{P}^{\mathsf{swi}} \rightarrow \mathcal{V}^{\mathsf{swi}}$
    1. If $\mathcal{V}(\mathsf{vk}, \mathtt{msg}, \sigma) \neq 1$ then abort, continue as follows otherwise.
    2. Run $\mathcal{P}$ on input x, w and ls^3 thus obtaining the 4th round ls^4 of LS.
    3. Send $((\mathtt{dec}, \mathsf{ls}^2), \mathsf{ls}^4)$ to $\mathcal{V}^{\mathsf{swi}}$.
5. $\mathcal{V}^{\mathsf{swi}}$: output 1 iff the following conditions are satisfied.
    1. Rec_{Σ} on input $\mathsf{vk}, \mathtt{com}, \mathtt{dec}, \mathsf{ls}^2$ accepts $(\mathsf{ls}^2, \mathtt{dec})$ as a decommitment of $\mathtt{com}$.
    2. $(\mathsf{ls}^1, \mathsf{ls}^2, \mathsf{ls}^3, \mathsf{ls}^4)$ is accepting for $\mathcal{V}$ with respect to the instance x.

Figure 5.2: Our SimWI AoK SWI from OWFs.

internally runs and interacts with a SWI prover $\mathcal{P}^{\mathsf{swi}}$ as $\mathcal{V}_i^{\mathsf{swi}}$ does, but acting as a proxy between $\mathcal{P}^{\mathsf{swi}\star}$ and ExtLS w.r.t. the messages of LS (for $i = 1, \dots, \mathsf{poly}(\lambda)$). The important observation is that E could fail if the following event NoExt happens with non-negligible probability: $\mathcal{P}^{\mathsf{swi}\star}$ opens the commitment $(\rho, \mathtt{com})$ to a different ls^2 during the rewinds. Indeed, in this case ExtLS could fail in obtaining a witness. We prove the following claim.

Claim 1. *There exists a negligible function ν such that* $\text{Prob}[\,\mathsf{NoExt}\,] < \nu(\lambda)$.

Proof. The proof is by contradiction, more specifically we now show an adversary $\mathcal{A}^{\Sigma}$ that extracts two signatures for two different messages in order to break the signature scheme Σ when $\text{Prob}[\,\mathsf{NoExt}\,]$ is non-negligible in λ.

If two decommitments of $(\mathtt{com}, \rho)$ w.r.t. two different messages ($\mathsf{ls}^{2'}$ and ls^2) are shown by $\mathcal{P}^{\mathsf{swi}\star}$ in the last round of SWI, $\mathcal{A}^{\Sigma}$ can extract two different signatures for two different messages by using the special binding of TC_{Σ}. More precisely, let vk be the verification key given by the challenger of the signature scheme, then our adversary $\mathcal{A}^{\Sigma}$ works as follows.

For all $i \in \{1, \dots, \mathsf{poly}(\lambda)\} - \{j\}$, $\mathcal{A}^{\Sigma}$ interacts in the i-th session against $\mathcal{P}^{\mathsf{swi}\star}$ as $\mathcal{V}_i^{\mathsf{swi}}$ would do. Instead in the j-th session $\mathcal{A}^{\Sigma}$ runs as E would do, using vk to compute the first round, and the oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ to compute a signature σ of a message m sent by $\mathcal{A}^{\Sigma}$ in the second round. Since we are assuming (by contradiction) that during the rewinds from the 4th round to the 3rd round the commitment $(\rho, \mathtt{com})$ (sent in the second round by $\mathcal{P}^{\mathsf{swi}\star}$) is opened in more than one way, then, by using the special binding of TC_{Σ}, $\mathcal{A}^{\Sigma}$ extracts and outputs two signatures for two different messages. We conclude this proof with the following two observations. First, the signature oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ is called only once since, by construction of E, the second round is played by $\mathcal{P}^{\mathsf{swi}\star}$ only once. Second, the extractor E is an expected polynomial-time algorithm while $\mathcal{A}^{\Sigma}$ must be a strict polynomial-time algorithm. This mean that the execution E has to be truncated. Obviously the running time of the extraction procedure can be truncated to a sufficiently long value so that with non-negligible probability the truncated extraction procedure will still yield the event NoExt to happened and this is sufficient for $\mathcal{A}^{\Sigma}$ to break the signature scheme[10]. $\square$

SimWI. In order to prove that SWI is SimWI (Definition 21) for the γ-prefix language L^{γ} we prove the following lemma.

Lemma 3. $\{\mathsf{wimim}_{\mathsf{SWI}}(1^{\lambda}, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^{\star}} \approx \{\mathsf{sim}_{\mathsf{SWI}}^{\mathcal{S}^{\mathsf{swi}}}(1^{\lambda}, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^{\star}}$.

Proof. Here we actually prove something stronger. Indeed we prove the security of SWI considering a MiM adversary $\mathcal{A}^{\mathsf{swi}}$ that has additional power both in the left and in the right sessions. More precisely in the left session $\mathcal{A}^{\mathsf{swi}}$ can choose the statement to be proved (and the related witness) in the third round. That is, in the last round that goes from $\mathcal{A}^{\mathsf{swi}}$ to $\mathcal{P}^{\mathsf{swi}}$.

Also, in all right sessions $\mathcal{A}$ chooses a family of statements $\mathcal{X}_w = \{x : (x, w) \in \mathsf{Rel}_{\mathsf{L}^{\gamma}} \text{ or } x \notin L^{\gamma}\}$ in the second round, and then adaptively picks the statement to be proved from that family in the last round. Roughly speaking $\mathcal{X}_w$ is the family of statements that either share the same witness w, or do not belong to the γ-prefix language L^{γ}. In this way the MiM adversary has the power to adaptively choosing the statement to be proved in the last round of every right session conditioned on $\mathcal{X}_w$, that has to be fixed in the second round. In the rest of the thesis we will refer to a SimWI protocol that is secure also in this setting as *adaptive-input* SimWI.

We start by showing the simulator $\mathcal{S}^{\mathsf{swi}}$ and giving an overview of the entire proof. The simulator is described in Figure 5.3. Roughly, $\mathcal{S}^{\mathsf{swi}}$ interacts against $\mathcal{A}^{\mathsf{swi}}$ in both the left and right sessions. In the left session $\mathcal{S}^{\mathsf{swi}}$ runs TFake_{Σ} to compute and send a commitment $\mathtt{com}$. $\mathcal{S}^{\mathsf{swi}}$ then rewinds $\mathcal{A}^{\mathsf{swi}}$ from the 3rd to the 2nd round, in order to obtain two valid signatures σ_1, σ_2 for two different messages $(\mathtt{msg}_1, \mathtt{msg}_2)$. This informations constitute the trapdoor $\mathtt{tk}$ for TC_{Σ}. After that $\mathtt{tk}$ is computed, $\mathcal{S}^{\mathsf{swi}}$ comes back to the main thread execution. Upon receiving ls^3 and x in the 3rd round from $\mathcal{A}^{\mathsf{swi}}$, $\mathcal{S}^{\mathsf{swi}}$ computes an accepting transcript for LS $(\mathsf{ls}^1, \mathsf{ls}^2, \mathsf{ls}^3, \mathsf{ls}^4)$ running the Special HVZK simulator of LS on input ls^1, received in the 1st round from $\mathcal{A}^{\mathsf{swi}}$, and (x, ls^3). In the last round computes, by using $\mathtt{tk}$, the decommitment information $(\mathtt{dec}, \mathsf{ls}^2)$ for $\mathtt{com}$, and sends $(\mathtt{dec}, \mathsf{ls}^2, \mathsf{ls}^4)$ to $\mathcal{A}^{\mathsf{swi}}$. In the i-th right session, for $i = 1, \dots, \mathsf{poly}(\lambda)$, $\mathcal{S}^{\mathsf{swi}}$ acts as $\mathcal{V}_i^{\mathsf{swi}}$ would do against $\mathcal{A}^{\mathsf{swi}}$. When the execution against $\mathcal{A}^{\mathsf{swi}}$ ends, $\mathcal{S}^{\mathsf{swi}}$ outputs the view of $\mathcal{A}^{\mathsf{swi}}$.

In the security proof we denote by $\{\mathsf{wimim}_{\mathcal{H}_i}(1^{\lambda}, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^{\star}}$ the random variable describing 1) the view of $\mathcal{A}^{\mathsf{swi}}$, 2) the output of $\mathcal{V}_i^{\mathsf{swi}}$ for $i = 1, \dots, \mathsf{poly}(\lambda)$, 3) the first γ-bits $\tilde{w}_1^{\gamma}, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}^{\gamma}$ of the corresponding witnesses $\tilde{w}_1, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}$ w.r.t. the instances $\tilde{x}_1, \dots, \tilde{x}_{\mathsf{poly}(\lambda)}$ that appear in $\mathcal{A}^{\mathsf{swi}}$'s view except that $\tilde{w}_i^{\gamma} = \perp$ if $\mathcal{V}_i^{\mathsf{swi}}$ rejected, with $i = 1, \dots, \mathsf{poly}(\lambda)$ [11].

[10] The same arguments are used in [GK96b]. The same standard argument about truncating the execution of an expected polynomial-time algorithm is used in another proofs but for simplicity we will not repeat this discussion.

[11] To ease the notation sometimes we will refer to $\{\mathsf{wimim}_{\mathcal{H}_i}(1^{\lambda}, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^{\star}}$ using just $\mathsf{wimim}_{\mathcal{H}_i}(1^{\lambda}, z)$.

Common input: security parameters λ and instance length ℓ.
Internal simulation of the left session:

1. Upon receiving $(\mathsf{vk}, \mathsf{ls}^1, \rho)$ from $\mathcal{A}^{\mathsf{swi}}$.
   1.1. Run TFake_Σ on input 1^λ, vk, ρ to compute the pair $(\mathtt{com}, \mathtt{aux})$.
   1.2. Pick a message $\mathtt{msg}_1 \leftarrow \{0,1\}^\lambda$.
   1.3. Send $(\mathtt{com}, \mathtt{msg}_1)$ to $\mathcal{A}^{\mathsf{swi}}$.
2. Upon receiving $(\mathsf{ls}^3, \sigma_1, x, \perp)$ from $\mathcal{A}^{\mathsf{swi}}$.
   2.1. If $\mathcal{V}(\mathsf{vk}, \mathtt{msg}_1, \sigma) \neq 1$ then abort, continue as follows otherwise.
   2.2. Repeat Step 1.3, 1.2 and follow-up right session message up to λ/p times[a] in order to obtain a signature σ_2 of a random message $\mathtt{msg}_2 \neq \mathtt{msg}_1$. Abort in case of failure in obtaining σ_2 in such λ/p attempts otherwise return to the main thread.
   2.3. Run the Special HVZK simulator of LS on input $(x, \mathsf{ls}^1, \mathsf{ls}^3)$ in order to obtain $(\mathsf{ls}^2, \mathsf{ls}^4)$.
   2.4. Run TFake_Σ on input $\mathtt{tk} = (\mathtt{msg}_1, \mathtt{msg}_2, \sigma_1, \sigma_2)$, vk, ρ, $\mathtt{com}$, $\mathtt{aux}$ and ls^2 to compute $\mathtt{dec}$.
   2.5. Send $((\mathtt{dec}, \mathsf{ls}^2), \mathsf{ls}^4)$ to $\mathcal{A}^{\mathsf{swi}}$.

Internal simulation of the right sessions:

1. For $i = 1, \dots, \mathsf{poly}(\lambda)$ acts as $\mathcal{V}_i^{\mathsf{swi}}$ would do against $\mathcal{A}^{\mathsf{swi}}$.

Output: When the execution against $\mathcal{A}^{\mathsf{swi}}$ ends, outputs the view of $\mathcal{A}^{\mathsf{swi}}$ and the bits $b_1, \dots, b_{\mathsf{poly}(\lambda)}$ where, for $i = 1, \dots, \mathsf{poly}(\lambda)$, $b_i = 0$ iff $\mathcal{V}_i^{\mathsf{swi}}$ is rejecting, $b_i = 1$ otherwise[b].

[a]We refer with p as the probability that $\mathcal{A}^{\mathsf{swi}}$ sends in the Step 2 a valid signature for a randomly chosen message.

[b]Of course, if $\mathcal{A}^{\mathsf{swi}}$ ends during the step 2.2 the simulator continues to work until that step is completed.

Figure 5.3: The SimWI $\mathcal{S}^{\mathsf{swi}}$ for SWI.

The proof makes use of the following main hybrid experiments.

– The 1st hybrid experiment is $\mathcal{H}_1(1^\lambda, z)$. In this hybrid in the left session $\mathcal{P}^{\mathsf{swi}}$ interacts with $\mathcal{A}^{\mathsf{swi}}$ in order to prove the validity of the instance x using the witness w, while in the right sessions $\mathcal{V}_i^{\mathsf{swi}}$ interacts with $\mathcal{A}^{\mathsf{swi}}$ for $i = 1, \dots, \mathsf{poly}(\lambda)$. We want to prove that in the i-th right session $\mathcal{A}^{\mathsf{swi}}$ does not prove any false instance $\tilde{x}_i$ for any $i = 1, \dots, \mathsf{poly}(\lambda)$[12]. This property follows immediately from the adaptive-input AoK of SWI. We observe that in this case it is crucial that SWI is adaptive-input AoK, because we are considering an adversary $\mathcal{A}^{\mathsf{swi}}$ that can choose the instance to be proved in the last round of every right session.

[12]When we refer to a *proved instance* $\tilde{x}_i$ we implicitly assume that $\mathcal{V}_i^{\mathsf{swi}}$ is accepting, with $i = 1, \dots, \mathsf{poly}(\lambda)$.

- The 2nd hybrid experiment is $\mathcal{H}_2(1^\lambda, z)$ and differs from $\mathcal{H}_1(1^\lambda, z)$ in the way the commitment $\mathtt{com}$ and the decommitment information $\mathtt{dec}$ are computed in the left session. More precisely, $\mathcal{P}^{\mathsf{swi}}$ runs TFake_Σ to compute a commitment $(\rho, \mathtt{com})$, and subsequently to compute a decommitment of $(\rho, \mathtt{com})$ to the value ls^2 (we remark that no trapdoor is needed to run TFake_Σ in order to compute $(\rho, \mathtt{com})$). In more details, this experiment rewinds the adversary $\mathcal{A}^{\mathsf{swi}}$ from the 3rd to the 2nd round of the left session to extract two signatures σ_1, σ_2 of two different messages $(\mathtt{msg}_1, \mathtt{msg}_2)$ and uses them as trapdoor to run TFake_Σ. The indistinguishability between $\mathsf{wimim}_{\mathcal{H}_1}(1^\lambda, z)$ and $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z)$ comes from the hiding and the trapdoorness of TC_Σ.

- The 3rd hybrid experiment is $\mathcal{H}_3(1^\lambda, z)$ and differs from $\mathcal{H}_2(1^\lambda, z)$ in the way the transcript for LS is computed. In more details the Special HVZK simulator $\mathcal{S}$ of LS is used to compute the messages ls^2 and ls^4 instead of using the honest procedure $\mathcal{P}^{\mathsf{swi}}$. The indistinguishability between $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z)$ and $\mathsf{wimim}_{\mathcal{H}_3}(1^\lambda, z)$ comes from the Special HVZK of LS. We observe that the security proof ends with this hybrid experiment because $\mathsf{wimim}_{\mathcal{H}_3}(1^\lambda, z) \equiv \mathsf{sim}^{\mathcal{S}^{\mathsf{swi}}}_{\mathsf{SWI}}(1^\lambda, z)$.

□

The formal proof of the Theorem 5 can be found in the Sec. 5.6.1.

5.4 4-Round Concurrent NM Commitment Scheme

Our construction makes use of an adaptive-input SimWI AoK $\mathsf{SWI} = (\mathcal{P}^{\mathsf{swi}}, \mathcal{V}^{\mathsf{swi}})$ combined with a weak NM commitment scheme Π_{wom}. For our propose we consider a weak NM commitment scheme that with overwhelming probability for any accepting well formed commitment can be opened to only one message. We recall that the weak NM commitment scheme of [GRRV14] enjoys this property when instantiated with Naor's commitment scheme [Nao91].

We now consider the following language L based on the weak NM commitment scheme $\Pi_{\mathsf{wom}} = (\mathsf{Sen}_{\mathsf{wom}}, \mathsf{Rec}_{\mathsf{wom}})$:

$$L = \Big\{(\tau, \mathtt{id}) : \exists\ (m, \mathtt{dec}) \text{ s.t. } \mathsf{Rec}_{\mathsf{wom}} \text{ on input } \\ (m, \mathtt{dec}, \mathtt{id}) \text{ accepts } m \text{ as decommitment of } \tau\Big\}$$

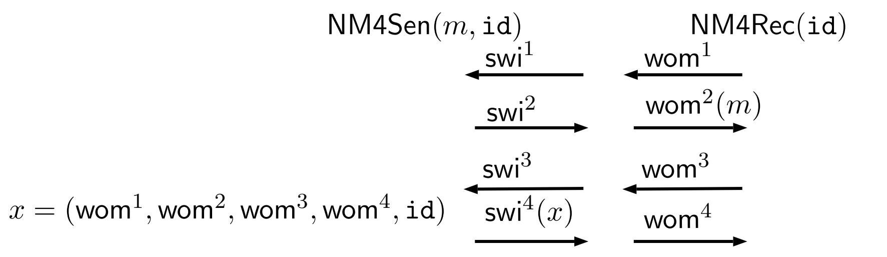
and the corresponding relation Rel_L.

We now use $\mathsf{SWI} = (\mathcal{P}^{\mathsf{swi}}, \mathcal{V}^{\mathsf{swi}})$ to upgrade a 4-round public-coin concurrent weak NM commitment scheme Π_{wom} with the property that after the second round there is at most one valid message, to a concurrent NM commitment scheme. We will be able to invoke the security of SWI, since the language L is a γ-prefix language with overwhelming probability with $\gamma = |m|$. In fact, given an instance $(\tau, \mathtt{id})$ of L all the witnesses $w_1, \ldots, w_n$ of $(\tau, \mathtt{id})$ have the form $m|\mathtt{dec}_i$ for $i = 1, \ldots, n$ (i.e. all witnesses have the same prefix m).

Consider a SimWI AoK for L. Let m be the message that $\mathsf{NM4Sen}$ wants to commit and $\mathtt{id}$ be the id for this session. The high-level idea of our protocol is depicted in Fig. 5.4.

In the 1st round the receiver $\mathsf{NM4Rec}$ computes and sends the 1st round swi^1 of SWI and the 1st round wom^1 of Π_{wom} using as input the $\mathtt{id}$. Then $\mathsf{NM4Sen}$ on input $\mathtt{id}$, the message m and the received 1st round, computes the 2nd round wom^2 of Π_{wom} in order to commit to the message m, using $\mathtt{id}$, furthermore he obtains $\mathtt{dec}_{\mathsf{wom}}$ s.t. $(m, \mathtt{dec}_{\mathsf{wom}})$ constitutes the decommitment information[13]. Moreover $\mathsf{NM4Sen}$ computes and sends the 2nd round swi^2 of SWI. In the

[13]In order to match the adaptive-input selection satisfied by SWI, message and randomness explaining the entire transcript are already fixed in this round.


$\mathsf{NM4Sen}(m, \mathtt{id})$
$\mathsf{NM4Rec}(\mathtt{id})$
swi^1
wom^1
swi^2
$\mathsf{wom}^2(m)$
swi^3
wom^3
$x = (\mathsf{wom}^1, \mathsf{wom}^2, \mathsf{wom}^3, \mathsf{wom}^4, \mathtt{id})$
$\mathsf{swi}^4(x)$
wom^4

- $t = (\mathsf{wom}^1, \mathsf{wom}^2(m), \mathsf{wom}^3, \mathsf{wom}^4)$ is the transcript generated from the execution of the weak non-malleable commitment Π_{wom} in which $\mathsf{Sen}_{\mathsf{wom}}$ wishes to commit to the message m (using id $\mathtt{id}$).
- $(\mathsf{swi}^1, \mathsf{swi}^2, \mathsf{swi}^3, \mathsf{swi}^4(x))$ in the transcript generated from an execution of the SimWI AoK SWI in which $\mathcal{P}^{\mathsf{swi}}$ proves that the commitment $(t, \mathtt{id})$ computed using Π_{wom} is well formed (i.e, it is a commitment of a message $m \neq \perp$).

Figure 5.4: 4-Round Concurrent NM Commitment Scheme from OWFs.

3rd round $\mathsf{NM4Rec}$ sends the 3rd round wom^3 of Π_{wom} and the 3rd round swi^3 of SWI. In the last round $\mathsf{NM4Sen}$ computes the 4th round wom^4 of Π_{wom}. Furthermore, $\mathsf{NM4Sen}$, using $(\mathsf{wom}^1, \mathsf{wom}^2, \mathsf{wom}^3, \mathsf{wom}^4, \mathtt{id})$ as instance and $(m, \mathtt{dec}_{\mathsf{wom}})$ as a witness, computes the 4th round swi^4 of SWI and sends $\mathsf{wom}^4, \mathsf{swi}^4$ to $\mathsf{NM4Rec}$. At this point $\mathsf{NM4Rec}$ accepts the commitment (i.e., the transcript of the protocol generated so far) iff the transcript for SWI is accepting for $\mathcal{V}^{\mathsf{swi}}$ with respect to the instance $(\mathsf{wom}^1, \mathsf{wom}^2, \mathsf{wom}^3, \mathsf{wom}^4, \mathtt{id})$. The decommitment phase of our scheme simply corresponds to the decommitment phase of Π_{wom}.

As described before, $\mathsf{SWI} = (\mathcal{P}^{\mathsf{swi}}, \mathcal{V}^{\mathsf{swi}})$ is used by $\mathsf{NM4Sen}$ to prove knowledge of a message and randomness consistent with the transcript computed using Π_{wom}. To execute SWI the instance is not needed until the last round.

The Fig. 5.5 describes in details our 4-round concurrent NM commitment scheme Π_{NM4Com}.

Theorem 6. *Assuming OWFs,* $\Pi_{\mathsf{NM4Com}} = (\mathsf{NM4Sen}, \mathsf{NM4Rec})$ *is a 4-round concurrent NM commitment scheme.*

The 4-round concurrent NM commitment scheme $\Pi_{\mathsf{NM4Com}} = (\mathsf{NM4Sen}, \mathsf{NM4Rec})$ relies on OWFs, because the adaptive-input SimWI AoK SWI can be constructed using OWFs only (see Theorem 5) and Π_{wom} can be instantiated using the weak concurrent non-malleable commitment scheme proposed in Section 5.5, that relies on OWFs. Note that the construction of Section 5.5 has also the additional property that we require (i.e. after the second round the only valid message and the corresponding decommitment informations are fixed).

The security proof is divided in two parts. In the 1st part we prove that Π_{NM4Com} is indeed a commitment scheme. In the second part we prove that Π_{NM4Com} is a one-many NM commitment scheme, and then we go from one-many to concurrent non-malleability by using Proposition 1.

Lemma 4. $\Pi_{\mathsf{NM4Com}} = (\mathsf{NM4Sen}, \mathsf{NM4Rec})$ *is a statistically binding computationally hiding commitment scheme.*

Proof. **Correctness.** The correctness follows directly from the delayed-input completeness of SWI and the correctness of Π_{wom}.

Common input: security parameter λ, instance length ℓ, $\mathsf{NM4Sen}$'s identity $\mathtt{id} \in \{0,1\}^\lambda$.
Input to $\mathsf{NM4Sen}$**:** $m \in \{0,1\}^{\mathsf{poly}\{\lambda\}}$.
Commitment phase:

1. $\mathsf{NM4Rec} \rightarrow \mathsf{NM4Sen}$
   1. Run $\mathsf{Rec}_{\mathsf{wom}}$ on input $1^\lambda, \mathtt{id}$ thus obtaining the 1st round wom^1 of Π_{wom}.
   2. Run $\mathcal{V}^{\mathsf{swi}}$ on input 1^λ and ℓ thus obtaining the 1st round swi^1 of SWI.
   3. Send $(\mathsf{swi}^1, \mathsf{wom}^1)$ to $\mathsf{NM4Sen}$.
2. $\mathsf{NM4Sen} \rightarrow \mathsf{NM4Rec}$
   1. Run $\mathcal{P}^{\mathsf{swi}}$ on input 1^λ, ℓ and swi^1 thus obtaining the 2nd round swi^2 of SWI.
   2. Run $\mathsf{Sen}_{\mathsf{wom}}$ on input $1^\lambda, \mathtt{id}, \mathsf{wom}^1$ and the message m thus obtaining the 2nd round wom^2 of Π_{wom} and $\mathtt{dec}_{\mathsf{wom}}$ s.t. $(m, \mathtt{dec}_{\mathsf{wom}})$ constitutes the decommitment information.
   3. Send $(\mathsf{swi}^2, \mathsf{wom}^2)$ to $\mathsf{NM4Rec}$.
3. $\mathsf{NM4Rec} \rightarrow \mathsf{NM4Sen}$
   1. Run $\mathsf{Rec}_{\mathsf{wom}}$ on input wom^2 thus obtaining the 3rd round wom^3 of Π_{wom}.
   2. Run $\mathcal{V}^{\mathsf{swi}}$ on input swi^2 thus obtaining the 3rd round swi^3 of SWI.
   3. Send $(\mathsf{wom}^3, \mathsf{swi}^3)$ to $\mathsf{NM4Sen}$.
4. $\mathsf{NM4Sen} \rightarrow \mathsf{NM4Rec}$
   1. Run $\mathsf{Sen}_{\mathsf{wom}}$ on input wom^3 thus obtaining the 4th round wom^4 of Π_{wom}.
   2. Set $x = (\mathsf{wom}^1, \mathsf{wom}^2, \mathsf{wom}^3, \mathsf{wom}^4, \mathtt{id})$ and $w = (m, \mathtt{dec}_{\mathsf{wom}})$ with $|x| = \ell$. Run $\mathcal{P}^{\mathsf{swi}}$ on input x, w and swi^3 thus obtaining the 4th round swi^4 of SWI.
   3. Send $(\mathsf{wom}^4, \mathsf{swi}^4)$ to $\mathsf{NM4Rec}$.
5. $\mathsf{NM4Rec}$: Set $x = (\mathsf{wom}^1, \mathsf{wom}^2, \mathsf{wom}^3, \mathsf{wom}^4, \mathtt{id})$ and accept the commitment iff $(\mathsf{swi}^1, \mathsf{swi}^2, \mathsf{swi}^3, \mathsf{swi}^4)$ is accepting for $\mathcal{V}^{\mathsf{swi}}$ with respect to the instance x.

Decommitment phase:

1. $\mathsf{NM4Sen} \rightarrow \mathsf{NM4Rec}$: Send $(m, \mathtt{dec}_{\mathsf{wom}})$ to $\mathsf{NM4Rec}$.
2. $\mathsf{NM4Rec}$: accept m as the committed message if and only if $\mathsf{Rec}_{\mathsf{wom}}$, on input $(m, \mathtt{dec}_{\mathsf{wom}})$, accepts m as the committed message of $(\mathsf{wom}^1, \mathsf{wom}^2, \mathsf{wom}^3, \mathsf{wom}^4, \mathtt{id})$.

Figure 5.5: Our 4-round concurrent NM commitment scheme Π_{NM4Com} from OWFs.

Statistically Binding. Observe that the message given in output in the decommitment phase of Π_{NM4Com} is the message committed using Π_{wom}. Moreover the decommitment of Π_{NM4Com} coincides with the decommitment of Π_{wom}. Since Π_{wom} is statistically binding then so is Π_{NM4Com}.

Computationally Hiding. Computational hiding follows immediately from Lemma 5.

□

Lemma 5. *For all* $m \in \{0,1\}^{\mathsf{poly}(\lambda)}$ $\{\mathsf{mim}_{\Pi_{\mathsf{NM4Com}}}^{\mathcal{A}_{\mathsf{NMCom}},m}(z)\}_{z\in\{0,1\}^\star} \approx \{\mathsf{sim}_{\Pi_{\mathsf{NM4Com}}}^{\mathsf{Sim}^{\mathsf{NM4Com}}}(1^\lambda, z)\}_{z\in\{0,1\}^\star}$.

We denote by $\{\mathsf{mim}_{\mathcal{H}_i^m}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z)\}_{z\in\{0,1\}^\star}$ the random variable describing the view of the MiM $\mathcal{A}^{\mathsf{NM4Com}}$ combined with the values that it commits in the the $\mathsf{poly}(\lambda)$ right sessions in hybrid $\mathcal{H}_i^m(z)$.

As required by the definition, we want to show that the distribution of the real game experiment (i.e., the view of the MiM $\mathcal{A}^{\mathsf{NM4Com}}$ when playing with $\mathsf{NM4Sen}$ committing m along with the messages committed in the right sessions) and the one of the output of a simulator are computationally indistinguishable. We start by showing the simulator $\mathsf{Sim}^{\mathsf{NM4Com}}$ and giving an overview of the entire proof. The simulator is described in Figure 5.6.

Common input: Security parameters: λ. $\mathsf{NM4Sen}$'s identity: $\mathtt{id} \in \{0,1\}^\lambda$.
Internal simulation of the left session:

1. Upon receiving $(\mathsf{swi}^1, \mathsf{wom}^1)$ from $\mathcal{A}^{\mathsf{NM4Com}}$.
   1.1. Run $\mathcal{P}^{\mathsf{swi}}$ on input 1^λ, ℓ and swi^1 thus obtaining the 2nd round swi^2 of SWI.
   1.2. Run $\mathsf{Sen}_{\mathsf{wom}}$ on input $1^\lambda, \mathtt{id}, \mathsf{wom}^1$ and the message 0^λ thus obtaining the 2nd round wom^2 of Π_{wom} and $\mathtt{dec}_{\mathsf{wom}}$ s.t. $(0^\lambda, \mathtt{dec}_{\mathsf{wom}})$ constitutes the decommitment informations.
   1.3. Send $(\mathsf{swi}^2, \mathsf{wom}^2)$ to $\mathcal{A}^{\mathsf{NM4Com}}$.
2. Upon receiving $(\mathsf{wom}^3, \mathsf{swi}^3)$ from $\mathcal{A}^{\mathsf{NM4Com}}$.
   2.1. Run $\mathsf{Sen}_{\mathsf{wom}}$ on input wom^3 thus obtaining the 4th round wom^4 of Π_{wom}.
   2.2. Set $x = (\mathsf{wom}^1, \mathsf{wom}^2, \mathsf{wom}^3, \mathsf{wom}^4, \mathtt{id})$ and $w = (0^\lambda, \mathtt{dec}_{\mathsf{wom}})$ with $|x| = \ell$. Run $\mathcal{P}^{\mathsf{swi}}$ on input x, w and swi^3 thus obtaining the 4th round swi^4 of SWI.
   2.3. Send $(\mathsf{wom}^4, \mathsf{swi}^4)$ to $\mathcal{A}^{\mathsf{NM4Com}}$.

Stand-alone commitment:

1. $\mathsf{Sim}^{\mathsf{NM4Com}}$ acts as a proxy between $\mathcal{A}^{\mathsf{NM4Com}}$ and $\mathsf{NM4Rec}_i$, with $i = 1, \dots, \mathsf{poly}(\lambda)$.

Figure 5.6: The simulator $\mathsf{Sim}^{\mathsf{NM4Com}}$ of Π_{NM4Com}.

– The 1st hybrid experiment is $\mathcal{H}_1^m(z)$. In this hybrid in the left session $\mathsf{NM4Sen}$ commits to m, while in the right sessions $\mathsf{NM4Rec}_i$ interacts with $\mathcal{A}^{\mathsf{NM4Com}}$ for $i = 1, \dots, \mathsf{poly}(\lambda)$.

We prove that in the i-th right session $\mathcal{A}^{\mathsf{NM4Com}}$ does not commit to a message $\tilde{m}_i = \perp$ for any $i = 1, \dots, \mathsf{poly}(\lambda)$. The proof follows immediately from the adaptive-input AoK of SWI. We observe that in this case it is crucial that SWI is adaptive-input AoK, because the theorem proved by $\mathcal{A}^{\mathsf{NM4Com}}$ are fully specified only in the last round of every right session. Clearly we have that $\mathsf{mim}^{\mathcal{A}^{\mathsf{NM4Com}},m}_{\Pi_{\mathsf{NM4Com}}}(z) = \mathsf{mim}^{\mathcal{A}^{\mathsf{NM4Com}},m}_{\mathcal{H}^m_1}(z)$.

– The 2nd hybrid experiment is $\mathcal{H}^m_2(z)$ and differs from $\mathcal{H}^m_1(z)$ in the way the transcript of SWI is computed. In this hybrid the simulator $\mathcal{S}^{\mathsf{swi}}$ of SWI is used to compute the transcript of SWI. The indistinguishability between $\mathsf{mim}^{\mathcal{A}^{\mathsf{NM4Com}},m}_{\mathcal{H}^m_1}(z)$ and $\mathsf{mim}^{\mathcal{A}^{\mathsf{NM4Com}},m}_{\mathcal{H}^m_2}(z)$ comes from the adaptive-input SimWI property of SWI. It is important to observe that we can properly rely on the adaptive-input SimWI property of SWI since the committed message in Π_{wom} is fixed in the second round. Therefore also the family of statement $\mathcal{X}_w = \{x : (x, w) \in \mathsf{Rel_L} \text{ or } x \notin L\}$ proved using SWI is implicitly fixed in the second round. Moreover we can rely on the security of SWI because the language L is a γ-prefix language for $\mathsf{prefix} = |m|$. Indeed, all witnesses of any instance of L have the same prefix (i.e., the committed message m). Therefore when using the simulator of SWI we are guaranteed that the distribution of the first γ bits of the witnesses corresponding to the statements proven by the adversary in the right sessions of SWI does not change. In turn, this implies that the distribution of the committed messages in the right sessions does not change since each message committed in a session is in the first γ bits of any witness corresponding to the statement proven in SWI in that session.

We also consider the hybrid experiments $\mathcal{H}^0_1(z)$, $\mathcal{H}^0_2(z)$, that are the same hybrid experiments described above with the difference that Π_{wom} is used to commit to a message 0^λ instead of m. From the same arguments described above we have that $\mathsf{mim}^{\mathcal{A}^{\mathsf{NM4Com}},m}_{\mathcal{H}^0_1}(z) \approx \mathsf{mim}^{\mathcal{A}^{\mathsf{NM4Com}},m}_{\mathcal{H}^0_2}(z)$ and that in the i-th right session of $\mathcal{H}^0_1(z)$ $\mathcal{A}^{\mathsf{NM4Com}}$ commits to a message $\tilde{m}_i = \perp$ with negligible probability (for any $i \in= 1, \dots, \mathsf{poly}(\lambda)$). We also observe that $\mathsf{mim}^{\mathcal{A}^{\mathsf{NM4Com}},m}_{\mathcal{H}^0_1}(z) = \mathsf{sim}^{\mathsf{Sim}^{\mathsf{NM4Com}}}_{\Pi_{\mathsf{NM4Com}}}(1^\lambda, z)$.

The only thing that remains to argue to complete the proof is that the view of $\mathcal{A}^{\mathsf{NM4Com}}$, along with messages committed in the right sessions of the execution of $\mathcal{H}^m_2(z)$, is indistinguishable from the view of $\mathcal{A}^{\mathsf{NM4Com}}$ along with the messages committed in the right sessions of $\mathcal{H}^0_2(z)$. This is actually ensured by the weak concurrent non-malleability of Π_{wom}. Indeed, from the arguments given above, in both $\mathcal{H}^m_2(z)$ and $\mathcal{H}^0_2(z)$ the adversary $\mathcal{A}^{\mathsf{NM4Com}}$ commits to a message $\tilde{m}_i = \perp$ with negligible probability for $i = 1, \dots, \mathsf{poly}(\lambda)$. Therefore we can use this $\mathcal{A}^{\mathsf{NM4Com}}$ to construct and adversary $\mathcal{A}^{\mathsf{wom}}$ that breaks the weak concurrent non-malleability of Π_{wom}. Roughly speaking, let $m, 0^\lambda$ be the challenge messages, then $\mathcal{A}^{\mathsf{wom}}$ works as following against the challenger $\mathcal{C}^{\mathsf{wom}}$. In the left session acts as a proxy for all the messages of Π_{wom} between $\mathcal{C}^{\mathsf{wom}}$ and $\mathcal{A}^{\mathsf{NM4Com}}$ and executes the simulator $\mathcal{S}^{\mathsf{swi}}$ of SWI in parallel. In the i-th right session $\mathcal{A}^{\mathsf{wom}}$ interacts as $\mathsf{Rec}_{\mathsf{wom},i}$ would do w.r.t. the messages of Π_{wom} and as $\mathcal{V}^{\mathsf{swi}}_i$ for the messages of SWI, for all $i = 1, \dots, \mathsf{poly}(\lambda)$. The distinguisher that break the concurrent weak non-malleability of Π_{wom} runs $\mathcal{D}^{\mathsf{NM4Com}}$ (that exists by contradiction) that distinguishes $\mathsf{mim}^{\mathcal{A}^{\mathsf{NM4Com}},m}_{\mathcal{H}^0_2}(z)$ from $\mathsf{mim}^{\mathcal{A}^{\mathsf{NM4Com}},m}_{\mathcal{H}^m_2}(z)$, and outputs what $\mathcal{D}^{\mathsf{NM4Com}}$ outputs.

A caveat that we have to address in this reduction is due to the rewinds made by $\mathcal{S}^{\mathsf{swi}}$ in the left session in order to compute the transcript of SWI. Indeed a rewind made in the left session could affect the reduction rewinding also the receivers of Π_{wom} involved in the reduction. More precisely could happen that in a session $j \in \{1, \dots, \mathsf{poly}(\lambda)\}$ the third round of Π_{wom} has to be played multiple times because of the multiple values $\tilde{\mathsf{wom}}^2_j$ received in the j-th right session. We can avoid this problem by sending a random string as a third round of Π_{wom}. In this way for the first value $\tilde{\mathsf{wom}}^2_j$ received from $\mathcal{A}^{\mathsf{NM4Com}}$ the reduction interacts with the receiver of Π_{wom} and for all the other values the reduction sends a random string. This is the reason why in our construction we require Π_{wom} to be public coin. One additional issue is the following. The simulator of SWI could rewind the entire left session during the reduction, therefore requiring to compute a new commitment of m for the protocol Π_{wom}. Since we are assuming that Π_{wom} is concurrent, the reduction can request to receive multiple commitments for the same message.

The formal proof can be found in Sec. 5.6.2.

5.5 On the Weak Concurrent Property of [GRRV14]

In this section we consider the one-one 8-round NM commitment (from OWFs) $\Pi_{\mathsf{1\text{-}1}}$ of [GRRV14] in order to show that one of the main tools used to construct $\Pi_{\mathsf{1\text{-}1}}$ is actually a weak concurrent public-coin non-malleable commitment (see Def. 20). In this section we will refer to the full version of [GRRV14].

$\Pi_{\mathsf{1\text{-}1}}$ is the result of the sequential execution of two protocols. The first one is a special 4-rounds public-coin commitment scheme Π_{wom}. The second one is a 4-round zero-knowledge AoK ZK that ensures that the commitment computed using Π_{wom} is well formed, that is, is not a commitment of a message $m = \perp$. We now prove that Π_{wom} is a weak concurrent NM commitment.

First of all it is easy to prove that if $\Pi_{\mathsf{1\text{-}1}}$ is hiding then Π_{wom} so is. We assume by contradiction that there exists an adversary $\mathcal{A}^h$ for the hiding of Π_{wom} and use it to break the hiding of $\Pi_{\mathsf{1\text{-}1}}$. The reduction simply acts as a proxy w.r.t. the messages of Π_{wom} between $\mathcal{A}^h$ and the challenge of the hiding of $\Pi_{\mathsf{1\text{-}1}}$, and acts a verifier w.r.t. the message of ZK. The output of the reduction corresponds to the output of $\mathcal{A}^h$.

Since the decommitment informations of $\Pi_{\mathsf{1\text{-}1}}$ correspond to the decommitment informations of Π_{wom}, and since $\Pi_{\mathsf{1\text{-}1}}$ is statistically binding, then Π_{wom} is statistically binding as well. In this section we consider the definition of *adaptive hiding*. The only differences with the classical definition of hiding is the following. Let m_0 and m_1 be the challenge messages. The challenger of adaptive hiding pick a random bit b and compute the commitment of m_b. The adversary, upon receiving the commitment, either outputs his guess $b' \in \{0,1\}$, or asks to receive another commitment of m_b (the latter step can be executed a polynomial number of times). The adversary is successful if $\text{Prob}[\, b = b' \,] - 1/2$ is non-negligible in the security parameter. It is easy to see that commitment scheme is hiding iff is adaptive hiding.

Now we are ready to prove the following lemma.

Lemma 6. *If OWFs exists then Π_{wom} is a 4-rounds public-coin concurrent weak NM commitment scheme.*

Proof. We start by proving that Π_{wom} is a 4-rounds public-coin one-many and then we go though one-many to concurrent by using similar arguments used in the proof of Proposition 1. Therefore now we show that for every PPT one-many MiM adversary $\mathcal{A}^{\mathsf{wom}}$ that does not commit to $\perp$ when a well formed commitment is received in the left session, there exists a PPT simulator S such that for every $m \in \{0,1\}^{\mathsf{poly}(\lambda)}$ the following ensembles are computationally indistinguishable: $\{\mathsf{mim}^{\mathcal{A},m}_{\Pi_{\mathsf{wom}}}(z)\}_{z\in\{0,1\}^\star} \approx \{\mathsf{sim}^{S}_{\Pi_{\mathsf{wom}}}(1^\lambda, z)\}_{z\in\{0,1\}^\star}$.

For sake of contradiction we suppose that there exists a MiM adversary $\mathcal{A}^{\mathsf{wom}}$ and a distinguisher $\mathcal{D}^{\mathsf{wom}}$ that distinguishes between the above two ensembles with probability at least $2p$, for some non-negligible p. We now use $\mathcal{A}^{\mathsf{wom}}$ and $\mathcal{D}^{\mathsf{wom}}$ to break the adaptive hiding of Π_{wom}. More precisely we construct an extractor E that extracts the messages committed by $\mathcal{A}^{\mathsf{wom}}$ in all the right sessions. Then we construct an adversary $\mathcal{A}^h$ that uses E in order to breaks the adaptive hiding of Π_{wom}. Before showing how exactly the reduction works, we formally describe the extractor E.

We start by assuming that only one right session is computed during the execution of Π_{wom}, and that the probability that $\mathcal{A}^{\mathsf{wom}}$ completes the execution of Π_{wom} is at least p. In this way we show an extractor that extracts the message committed in that session with probability greater than $1-p$ (that will be enough to break the hiding of Π_{wom}). Then, by using standard arguments (the same used in [Goy11]) we show that the extractor works properly also in the case of $\mathsf{poly}(\lambda)$ right sessions. The extractor that we use is the same used in Figure 3 of [GRRV14], but our purpose the only things to know about how E works are the following.

1. E takes as input a *well formed* transcript T of an execution of Π_{wom}, and has oracle access to the MiM adversary. A transcript T is well formed if both the commitment computed in the left session and the commitment computed in the right session are correctly computed; that is, they are commitments of messages different from $\perp$.
2. The extraction procedure of E rewinds $\mathcal{A}^{\mathsf{wom}}$, in the right session, from the 4th to the 2nd round $\mathsf{poly}(\lambda)$ times, feeding the MiM adversary with a random 3rd round every time.
3. If the rewinds made in the right session require the 4th message of the left session to be played again, E answers by using a specific strategy.

Let's now focus on the item 1. As we described before, in the non-malleable commitment scheme $\Pi_{\mathsf{1\text{-}1}}$ of [GRRV14] a ZK argument is executed in order to ensure that the commitment computed with Π_{wom} is well formed. We observe that, by definition of weak non-malleable commitment, we are considering now a MiM adversary that never commits to $\perp$ in the right session if the commitment that he receives in the left session is well formed. For that reason we can properly use the extractor E (slightly modified as we will describe later) according to the Theorem 2 of [GRRV14], that is the following.

Theorem 7 ([GRRV14, Theorem 2])**.** *Let T be the well formed transcript that is given as input to E. Let $\tilde{m}$ be MiM's commitment in the right session of T. Then the probability that E outputs a message different from $\tilde{m}$ is at most p.*

Let's now see more formally how we can use E to construct an adversary $\mathcal{A}^h$ for the adaptive hiding of Π_{wom}. $\mathcal{A}^h$ picks m and 0^λ as challenge messages and sends them to the challenger of the adaptive hiding of Π_{wom} $\mathcal{C}^h$. Then he acts as a proxy between $\mathcal{C}^h$ and $\mathcal{A}^{\mathsf{wom}}$ for all the messages of Π_{wom} of the left session. Instead, in the right session, $\mathcal{A}^h$ interacts against $\mathcal{A}^{\mathsf{wom}}$ as the receiver of Π_{wom} would do. When the execution ends, a transcript T for the execution of Π_{wom} is generated. At this point $\mathcal{A}^h$ runs the extractor E on input T in order to extract the messages committed in the right sessions. Then he feeds the distinguisher $\mathcal{D}^{\mathsf{wom}}$ with the extracted messages and the view of $\mathcal{A}^{\mathsf{wom}}$, and outputs what $\mathcal{D}^{\mathsf{wom}}$ outputs.

It is important to observe that the extractor E could rewind the left session, that is, could rewind the $\mathcal{C}^h$. There are two messages scheduling that can cause this problem. The synchronized scheduling, and the scheduling that require the entire left session to be played again (see Fig. 5.7 for an example of such scenario).

As described earlier in the item 2, E knows how to deal with the synchronized case. Indeed in that scheduling, because of how E works, can only happen that 4th message of the left session is required to be played again (in order to answer to a new 3rd message). For the second case the behavior of E is not well defined, indeed in that case E has to compute another commitment of m_b without knowing m_b. This is not a problem since we are making a reduction to the adaptive hiding of Π_{wom}. Therefore, when E has to play again the entire left session in order to commit to m_b, $\mathcal{A}^h$ can ask to $\mathcal{C}^h$ to start another execution of Π_{wom} and acts as a proxy between $\mathcal{C}^h$ and E. We recall that the adversary has the power to ask multiple commitments for the same challenge message because $\mathcal{C}^h$ is the challenger of adaptive hiding. In this case the extractor is still able to extract $\tilde{m}$ from the right session since, in every rewind, $\mathcal{A}^{\mathsf{wom}}$ receives well formed commitment on the left. Therefore, by the definition of weak non-malleable commitment, $\mathcal{A}^{\mathsf{wom}}$ is computing well formed commitment in the right session. This ensure that $\tilde{m}$ is extracted with non-negligible probability (by construction of E).

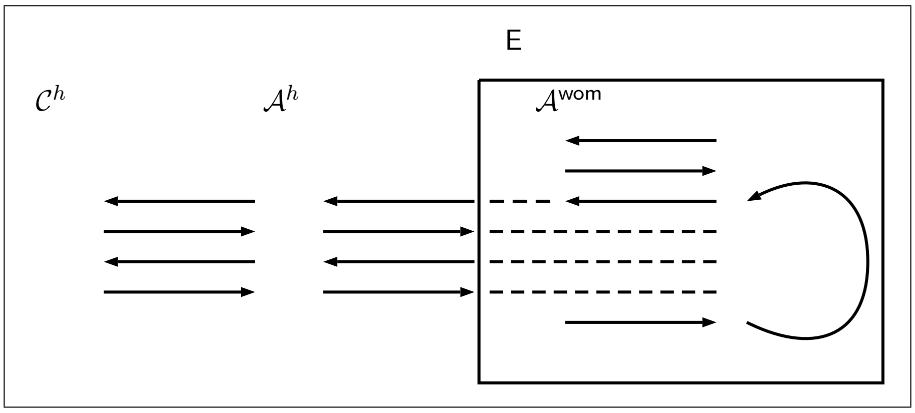


Figure 5.7: During the reduction to the adaptive hiding of Π_{wom} could be required to play the left session multiple times because of the rewinds made by E. In general this happen when it is required to recompute the second round of the left session, due to a new first round received by $\mathcal{A}^{\mathsf{wom}}$ (in the left session).

We now consider a MiM adversary that opens $\mathsf{poly}(\lambda)$ right sessions. First we point out that the extractor we construct is able to extract $\tilde{m}$ from the right interaction with high probability, without rewinding the left execution.

Therefore we can follow [Goy11], and run the extractor E one by one for all the right session. More precisely, for every right session $i = 1, \dots, \mathsf{poly}(\lambda)$:

- define a machine $\mathcal{M}_i$ with emulates all the right sessions excepts session i on its own and expose the i-th session to an outside receiver;
- run the extractor on the machine $\mathcal{M}_i$ giving it as input the left view as in the main thread and the right view of the i-th session in the main thread.

As for [Goy11], by the union bound our extractor will succeed in extracting from all of the right interactions with high probability.

In order to conclude the proof we now prove the following claim.

Claim 2. *Let Π_{1-n} be a weak one-many non-malleable, then Π_{1-n} is also weak concurrent non-malleable commitment.*

Proof. The security proof is similar to the security proof of Proposition 1 that is provided in [LPV08], but for sake of completeness here we propose a full security proof of our claim.

Let $\mathcal{A}$ be a weak man-in-the-middle adversary that participates in at most $v = \mathsf{poly}(\lambda)$ concurrent executions. We provide a concurrent simulator S for $\mathcal{A}$. S proceeds as follows on input 1^λ and z. S incorporates $\mathcal{A}$ and internally emulates all the left interactions for $\mathcal{A}$ by simply honestly committing to the string 0^λ. Messages from the right interactions are instead forwarded externally to the receivers of Π_{1-n}. Finally S outputs the view of $\mathcal{A}$. We show that the values that S commits to are indistinguishable from the values that $\mathcal{A}$ commits to. Suppose, for contradiction, that this is not the case. Then, there exists a polynomial-time distinguisher $\mathcal{D}$ and a polynomial $p(\lambda)$ such that for infinitely many λ, there exist messages $m_0, m_1, \dots, m_v$ and $z \in \{0,1\}^\star$ s.t. the distinguisher $\mathcal{D}$ distinguishes between $\mathsf{mim}_{\Pi_{1-n}}^{\mathcal{A}, m_1, \dots, m_v}(z)$ and $\mathsf{sim}_{\Pi_{1-n}}^{S}(1^\lambda, z)$ with probability $1/p(\lambda)$. Fix a generic λ for which this happens. Consider the hybrid simulator S_i that on input 1^λ, $z' = m_1, \dots, m_v, z$, proceeds just as S, with the exception that in left interactions $j \leq i$, it instead commits to m_j. It directly follows that $\mathsf{mim}^{\mathcal{A}, m_1, \dots, m_v}_{\Pi_{1-n}}(z) = \mathsf{sim}^{S_v}_{\Pi_{1-n}}(1^\lambda, z')$ and $\mathsf{sim}^{S}_{\Pi_{1-n}}(1^\lambda, z) = \mathsf{sim}^{S_0}_{\Pi_{1-n}}(1^\lambda, z')$. By a standard hybrid argument there exists an $i \in \{1, \dots, v\}$ such that

$$\begin{aligned}&\text{Prob}\left[\, a \leftarrow \mathsf{sim}^{S_{i-1}}_{\Pi_{1-n}}(1^\lambda, z') : \mathcal{D}(1^\lambda, z', a) = 1 \,\right] - \\ &\text{Prob}\left[\, b \leftarrow \mathsf{sim}^{S_i}_{\Pi_{1-n}}(1^\lambda, z') : \mathcal{D}(1^\lambda, z', b) = 1 \,\right] \geq 1/(p(\lambda)v).\end{aligned}$$

Note that the only difference between the executions by $S_{i-1}(1^\lambda, z')$ and $S_i(1^\lambda, z')$ is that in the former $\mathcal{A}$ receives a commitment to 0^λ in session i, whereas in the latter it receives a commitment to m_i. Consider the one-many adversary $\tilde{\mathcal{A}}$ that on input $\tilde{z} = z', \lambda, i$ executes $S_{i-1}(1^\lambda, z')$ with the exception that the i-th left interaction is forwarded externally. Consider, the function $\mathsf{reconstruct}$ that on input $\mathsf{mim}^{\tilde{\mathcal{A}}, 0^\lambda}_{\Pi_{1-n}}(z)$ i.e. values $\tilde{m}'_1, \dots, \tilde{m}'_v$ and the view of $\tilde{A}$, reconstructs the view view of $\mathcal{A}$ in the emulation by $\tilde{\mathcal{A}}$, and outputs $\tilde{m}'_1, \dots, \tilde{m}'_v, \mathsf{view}$. By construction, it follows that

$$\mathsf{reconstruct}(\mathsf{mim}^{\tilde{\mathcal{A}}, 0^\lambda}_{\Pi_{1-n}}(z)) = \mathsf{sim}^{S_{i-1}}_{\Pi_{1-n}}(1^\lambda, z')$$

$$\mathsf{reconstruct}(\mathsf{mim}^{\tilde{\mathcal{A}}, m_i}_{\Pi_{1-n}}(z)) = \mathsf{sim}^{S_i}_{\Pi_{1-n}}(1^\lambda, z')$$

Since $\mathsf{reconstruct}$ is polynomial-time computable, this contradicts the one-many non-malleability of Π_{1-n}. $\square$

$\square$

A useful property enjoyed by Π_{wom}, that is also used in this work, is that the decommitment informations (dec, m) are fixed in the second round of the Π_{wom}. We also observe that if the underling commitment scheme (the one used in the construction of Π_{wom}) is perfectly binding, so is Π_{wom}. We recall that the existence of 1-to-1 OWFs implies the existence of perfectly binding commitment scheme. Therefore we can consider an instantiation of the underling commitment of Π_{wom} using 1-to-1 OWFs, obtaining a 3-rounds perfectly binding public-coin concurrent weak NM commitment scheme. The number of rounds goes from 4 to 3 because in Π_{wom} the first round from receiver to sender is just related the underling commitment from OWFs (e.g. Naor's commitment).

5.6 Formal Proofs

5.6.1 Formal Proof of Th. 5

In this section we give more details about the security proof. In order to simplify the security proof, here we actually consider the notion of *adaptive Special HVZK* and *adaptive trapdoorness* instead of Special HVZK and trapdoorness. The only differences with the classical definition of Special HVZK is the following. Let $(\mathsf{ls}^1, \mathsf{ls}^3, x)$ be a challenge. The challenger of adaptive Special HVZK picks a random bit b and compute an accepting transcript $t = (\mathsf{ls}^1, \mathsf{ls}^2, \mathsf{ls}^3, \mathsf{ls}^4)$ for x. If $b = 0$ then t has been computed by using $\mathcal{P}$, with the Special HVZK simulator otherwise. The adversary, upon receiving t, either outputs his guess $b' \in \{0, 1\}$, or asks to receive another transcript t according to a new challenge $(\mathsf{ls}^{1'}, \mathsf{ls}^{3'}, x')$. Note that the adversary can require a polynomial number of transcripts according to different challenges before he outputs b'. The adversary is successful if $\text{Prob}\left[\, b = b' \,\right] - 1/2$ is non-negligible in the security parameter. It is easy to see that a protocol is Special HVZK iff is adaptive Special HVZK.

Analogously, the only differences with the definition of trapdoorness given in Definition 6 is the following. Let (x, ρ, m) and $\mathtt{tk}$ s.t. $(x, \mathtt{tk}) \in \mathsf{Rel}_{L_\Sigma}$ be the challenge messages used by the adversary to interacts with the challenger. The challenger of adaptive trapdoorness picks a random bit b and compute a commitment $\mathtt{com}$ and the respectful decommitment information $(m, \mathtt{dec})$. If $b = 0$ then both $\mathtt{com}$ and $(m, \mathtt{dec})$ are computed using the trapdoor procedure TFake_Σ, using the honest procedure Sen_Σ otherwise. The adversary, upon receiving $\mathtt{com}$ and $(m, \mathtt{dec})$, either outputs his guess $b' \in \{0, 1\}$, or asks to start another interaction against the challenger by using different (x', ρ', m') and $\mathtt{tk}'$. Note that the adversary can start (in a sequential way) a polynomial number of other interactions using different (x', ρ', m') before he outputs b'. The adversary is successful if $\text{Prob}[\, b = b' \,] - 1/2$ is non-negligible in the security parameter. It is easy to see that commitment scheme enjoys the adaptive trapdoorness property iff enjoys the trapdoor property.

Below we formally describe both the hybrid experiments and the reductions involved in the security proof.

– As described before, the 1st hybrid experiment is $\mathcal{H}_1(1^\lambda, z)$. In this hybrid in the left session $\mathcal{P}^{\mathsf{swi}}$ interacts with $\mathcal{A}^{\mathsf{swi}}$ in order to prove the validity of the instance x (adaptively chosen by $\mathcal{A}^{\mathsf{swi}}$) using the witness w, while in the right sessions $\mathcal{V}_i^{\mathsf{swi}}$ interacts with $\mathcal{A}^{\mathsf{swi}}$ for $i = 1, \ldots, \mathsf{poly}(\lambda)$. Also, we have already observed that in every right session the probability that $\mathcal{A}^{\mathsf{swi}}$ proves a false instance is negligible because of the property of adaptive-input AoK of SWI.

– We now consider an intermediate hybrid experiment $\mathcal{H}_1^{\mathsf{rew}}(1^\lambda, z)$ that differs from $\mathcal{H}_1(1^\lambda, z)$ as follows. In the left session, by rewinding the adversary $\mathcal{A}^{\mathsf{swi}}$ from the 3rd to the 2nd round, two signatures σ_1, σ_2 for two distinct messages $(\mathtt{msg}_1, \mathtt{msg}_2)$ are extracted. First of all we prove that in $\mathcal{H}_1^{\mathsf{rew}}(1^\lambda, z)$ two signatures are obtained with non-negligible probability. Let p the probability that a valid signature σ for the message $\mathtt{msg}$ is provided by $\mathcal{A}^{\mathsf{swi}}$, that is $\mathcal{V}(\mathsf{vk}, \mathtt{msg}, \sigma) = 1$. If a valid signature is received, then the steps 2.3, 2.4, and follow-up right session messages, are repeated up to λ/p times. This implies that for a non-negligible p the probability that $\mathcal{A}^{\mathsf{swi}}$ does not give a 2nd valid signature for two randomly chosen messages after λ/p rewinds is negligible in λ. For that reason the above deviation increases the abort probability of the experiment only by a negligible amount, therefore we can claim that $\mathsf{wimim}_{\mathcal{H}_1}(1^\lambda, z) \approx \mathsf{wimim}_{\mathcal{H}_1^{\mathsf{rew}}}(1^\lambda, z)$.

– We now consider the 2nd hybrid experiment $\mathcal{H}_2(1^\lambda, z)$ that differs from $\mathcal{H}_1^{\mathsf{rew}}(1^\lambda, z)$ as follows. The value $\mathtt{com}$ is computed by running TFake_Σ instead of Sen_Σ. We recall that by Definition 6 no trapdoor information is required in order to compute $\mathtt{com}$ using the algorithm TFake_Σ. Then the hybrid experiment proceeds as $\mathcal{H}_1^{\mathsf{rew}}(1^\lambda, z)$ until signatures (σ_1, σ_2) for two different messages $(\mathtt{msg}_1, \mathtt{msg}_2)$ are extracted. At this point the extracted information is used as trapdoor to run TFake_Σ in order to compute a decommitment $(\mathtt{dec}, \mathsf{ls}^2)$ w.r.t. the commitment $(\rho, \mathtt{com})$. First, we prove that the probability of extracting two valid signatures for two different messages in this hybrid experiment is greater (or equal) than the probability that this event occurs in $\mathcal{H}_1^{\mathsf{rew}}(1^\lambda, z)$, and then we prove that $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z) \approx \mathsf{wimim}_{\mathcal{H}_1^{\mathsf{rew}}}(1^\lambda, z)$. The following claim is sufficient to prove the first part.

Claim 3. *Let $p_1(\lambda)$ be the (non-negligible) probability that a valid signature for $\mathtt{msg}$ is sent in the 3rd round of $\mathcal{H}_1^{\mathsf{rew}}(1^\lambda, z)$ by $\mathcal{A}^{\mathsf{swi}}$, and let $p_2(\lambda)$ be the probability that event occurs in $\mathcal{H}_2(1^\lambda, z)$ then, for every polynomial $\mathsf{poly}(\cdot)$, $p_1(\lambda) \leq p_2(\lambda) + 1/\mathsf{poly}(\lambda)$.*

Proof. Suppose by contradiction that there exists polynomial $\mathsf{poly}(\cdot)$ s.t., for a sufficiently large security parameter λ, $p_1(\lambda) > p_2(\lambda) + 1/\mathsf{poly}(\lambda)$, then we can construct an adversary

$\mathcal{A}^{\mathsf{TC}_\Sigma}$ that has non-negligible advantage in breaking the hiding of TC_Σ. Formally $\mathcal{A}^{\mathsf{TC}_\Sigma}$, interacting with the challenger for the hiding of TC_Σ, works as following

1. Upon receiving the 1st round from $\mathcal{A}^{\mathsf{swi}}$ in the left session, $\mathcal{A}^{\mathsf{TC}_\Sigma}$ computes ls^2 and sends it as the challenge message together with vk and ρ.
2. $\mathcal{A}^{\mathsf{TC}_\Sigma}$, upon receiving $\mathtt{com}$ from the challenger, uses it to compute and send the 2nd round of SWI to $\mathcal{A}^{\mathsf{swi}}$ on the left session.
3. Let $\overline{p}(\lambda)$ be the probability that a valid signature w.r.t. to vk is received in the third message of the left session by $\mathcal{A}^{\mathsf{swi}}$.
4. If $\overline{p}(\lambda)$ is negligible close to $p_1(\lambda)$ then outputs 0, outputs 1 otherwise[14].

We conclude the security proof by observing that the probability of $\mathcal{A}^{\mathsf{swi}}$ to send a pair $(x, w) \in \mathsf{Rel}_L$ in the 3rd round of $\mathcal{H}_1^{\mathsf{rew}}(1^\lambda, z)$ is negligibly close to the probability to the probability of receiving a pair $(x, w) \in \mathsf{Rel}_L$ in the 3rd round of $\mathcal{H}_2(1^\lambda, z)$. If it is not the case, then $\mathcal{A}^{\mathsf{swi}}$ can be used to break the hiding of TC_Σ. $\square$

We recall that the transcript of an execution of $\mathcal{H}_2(1^\lambda, z)$ differs from $\mathcal{H}_1^{\mathsf{rew}}(1^\lambda, z)$ in the way the commitment $(\rho, \mathtt{com})$ of TC_Σ and the corresponding decommitment information are computed. To prove that $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z) \approx \mathsf{wimim}_{\mathcal{H}_1^{\mathsf{rew}}}(1^\lambda, z)$ we proceed by contradiction constructing an adversary $\mathcal{A}^{\mathsf{TC}_\Sigma}$ for the adaptive trapdoorness property of the 2-round instance-dependent trapdoor commitment scheme TC_Σ. Roughly speaking, we are assuming by contradiction that there exists an adversary $\mathcal{A}^{\mathsf{swi}}$ and a distinguisher $\mathcal{D}^{\mathsf{swi}}$ that distinguishes $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z)$ from $\mathsf{wimim}_{\mathcal{H}_1^{\mathsf{rew}}}(1^\lambda, z)$. Therefore, we can construct an adversary $\mathcal{A}^{\mathsf{TC}_\Sigma}$ that interacts with $\mathcal{A}^{\mathsf{swi}}$ in the left and the right sessions according to both $\mathcal{H}_1^{\mathsf{rew}}(1^\lambda, z)$ and $\mathcal{H}_2(1^\lambda, z)$ for all messages except for the messages of TC_Σ. For these messages $\mathcal{A}^{\mathsf{TC}_\Sigma}$ acts as a proxy between $\mathcal{A}^{\mathsf{swi}}$ and the challenger (involved in the adaptive-trapdoor security game) in the left session.

Let $\tilde{x}_1, \dots, \tilde{x}_{\mathsf{poly}(\lambda)}$ be the statements proved in the right sessions by $\mathcal{A}^{\mathsf{swi}}$. $\mathcal{A}^{\mathsf{TC}_\Sigma}$ extracts from all the right sessions the witnesses $\tilde{w}_1, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}$ for $\tilde{x}_1, \dots, \tilde{x}_{\mathsf{poly}(\lambda)}$ relying on the adaptive-input PoK property of LS. Then $\mathcal{A}^{\mathsf{TC}_\Sigma}$ runs the distinguisher $\mathcal{D}^{\mathsf{swi}}$ on input the first γ-bits of each witnesses, namely $\tilde{w}_1^\gamma, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}^\gamma$, and outputs what $\mathcal{D}^{\mathsf{swi}}$ outputs.

Formally, against the challenger of adaptive trapdoorness $\mathcal{C}^{\mathsf{TC}_\Sigma}$, $\mathcal{A}^{\mathsf{TC}_\Sigma}$ works as following.

1. Upon receiving the 1st round from $\mathcal{A}^{\mathsf{swi}}$, $\mathcal{A}^{\mathsf{TC}_\Sigma}$ computes ls^2 and sends it as the challenge message together with ρ and vk to $\mathcal{C}^{\mathsf{TC}_\Sigma}$.
2. $\mathcal{A}^{\mathsf{TC}_\Sigma}$, upon receiving $\mathtt{com}$ from $\mathcal{C}^{\mathsf{TC}_\Sigma}$, uses it to compute and send the 2nd round of SWI to $\mathcal{A}^{\mathsf{swi}}$ on the left.
3. $\mathcal{A}^{\mathsf{TC}_\Sigma}$ extracts two valid signatures of two different messages from the left session therefore obtaining the trapdoor $\mathtt{tk}$ for TC_Σ. Then $\mathcal{A}^{\mathsf{TC}_\Sigma}$ sends $\mathtt{tk}$ to $\mathcal{C}^{\mathsf{TC}_\Sigma}$.
4. Upon receiving $\mathtt{dec}$ from $\mathcal{C}^{\mathsf{TC}_\Sigma}$, $\mathcal{A}^{\mathsf{TC}_\Sigma}$ uses $\mathtt{dec}$ to complete the left session against $\mathcal{A}^{\mathsf{swi}}$.
5. $\mathcal{A}^{\mathsf{TC}_\Sigma}$ uses the extractor of LS to extract from the $\mathsf{poly}(\lambda)$ right sessions the witnesses used by $\mathcal{A}^{\mathsf{swi}}$ to compute the LS PoKs (the witnesses correspond to statements $\tilde{x}_i$ proved by $\mathcal{A}^{\mathsf{swi}}$ in the i-th right session, for $i = 1, \dots, \mathsf{poly}(\lambda)$). We recall that the extractor of LS succeeds with non-negligible probability from the adaptive-input AoK property of LS. Indeed, since we have proved in the previous hybrid experiment that

[14]The output values are computed according to Definition 6.

event NoExt happens with negligible probability, so is in this hybrid experiment. If it is not the case then we can construct another adversary that breaks the adaptive trapdoorness property of $\mathsf{TC}_{\mathbf{\Sigma}}$.

6. From what we argue above we can assume that the extraction succeeds with non-negligible probability, and thus $\mathcal{A}^{\mathsf{TC}_{\Sigma}}$ can run the distinguisher $\mathcal{D}^{\mathsf{swi}}$ on input the first γ-bits of each extracted witnesses to distinguish $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z)$ from $\mathsf{wimim}_{\mathcal{H}_1^{\mathsf{rew}}}(1^\lambda, z)$, thus breaking the trapdoorness property of $\mathsf{TC}_{\mathbf{\Sigma}}$.

We now argue that rewinds made on the right sessions (during the step 5) do not affect the reduction. We distinguish between three kinds of right sessions: $\mathsf{3Rewind}$, $\mathsf{AllRewind}$ and $\mathsf{NoRewind}$.

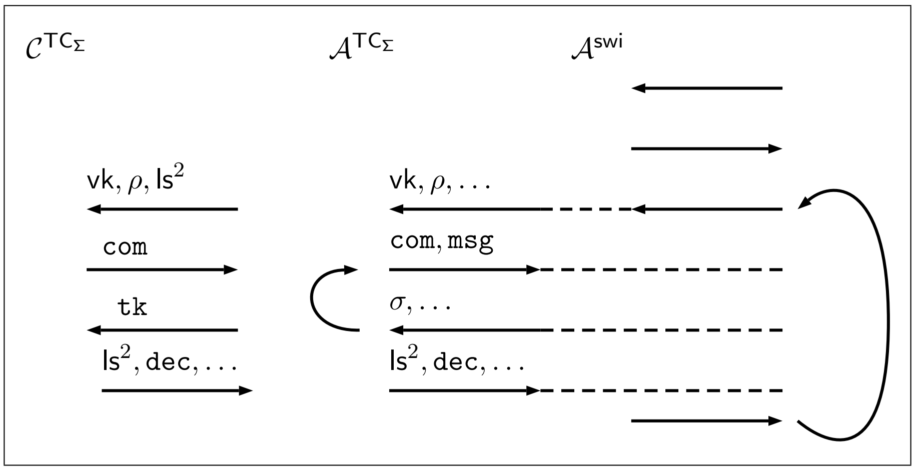
– A right session is $\mathsf{3Rewind}$ if during the extraction procedure is required, in the left session, to play again *only* the 4th round having received a different 3rd round (i.e., a different ls^3).
– A right session is $\mathsf{AllRewind}$ if during the extraction procedure is required, in the left session, to play again *the entire* left session due to a different first round $(\mathsf{vk}', \mathsf{ls}^{1\prime}, \rho')$ received from $\mathcal{A}^{\mathsf{swi}}$.
– A right session is $\mathsf{NoRewind}$ if it is neither $\mathsf{3Rewind}$ nor $\mathsf{AllRewind}$.

It is easy to see that $\mathsf{NoRewind}$ right sessions do not interfere with the reduction, that is, the left session is not actually rewound because both the first and the third round stay the same during the extraction. It is less trivial to deal with the $\mathsf{3Rewind}$ and $\mathsf{AllRewind}$ sessions but still, that sessions do not cause issue during the reduction. When the rewinds occur in the $\mathsf{3Rewind}$ right session, to compute the 4th round it is possible to reuse $(\mathtt{dec}, \mathsf{ls}^2)$ in order to compute an accepting transcript for LS $(\mathsf{ls}^1, \mathsf{ls}^2, \mathsf{ls}^{3\prime}, \mathsf{ls}^{4\prime})$ w.r.t. $\mathsf{ls}^{3\prime}$ and a (potentially different) statement x' that is provided in the 3rd round by $\mathcal{A}^{\mathsf{swi}}$. This is possible because of the delayed-input completeness property enjoyed by LS and because the witness for the theorem x' is provided by $\mathcal{A}^{\mathsf{swi}}$ in the third round of the left session. For the $\mathsf{AllRewind}$ right sessions, the adversary $\mathcal{A}^{\mathsf{TC}_{\Sigma}}$, upon receiving a new first round $(\mathsf{vk}', \mathsf{ls}^{1\prime}, \rho')$ from $\mathcal{A}^{\mathsf{swi}}$, starts a new interaction against the challenger of adaptive trapdoorness executing all steps described above starting starting from step 1 as summarized in Figure 5.8.

– The 3rd hybrid experiment is $\mathcal{H}_3(1^\lambda, z)$ and differs from $\mathcal{H}_2(1^\lambda, z)$ in the way the transcript for LS is computed. More precisely, the Special HVZK simulator $\mathcal{S}$ of LS is used to compute the messages ls^2 and ls^4 instead of using the honest prover procedure $\mathcal{P}$. We observe that this is possible because in this hybrid experiment the commitment $(\rho, \mathtt{com})$ can be opened to any value ls^2 due to the trapdoor property of $\mathsf{TC}_{\mathbf{\Sigma}}$. Therefore, after receiving ls^1 in the 1st round, and (ls^3, x, w) in the 3rd round from $\mathcal{A}^{\mathsf{swi}}$, $(\mathsf{ls}^1, \mathsf{ls}^3, x)$ are given as input of $\mathcal{S}$ that will output $\mathsf{ls}^2, \mathsf{ls}^4$. These values are then used to compute the 4th round of the left session.

Before proving that $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z) \approx \mathsf{wimim}_{\mathcal{H}_3}(1^\lambda, z)$ we observe that the first three round of the left interaction in $\mathcal{H}_3(1^\lambda, z)$ are identically distributed to the first three round of $\mathcal{H}_2(1^\lambda, z)$. Therefore the probability of receiving a pair $(x, w) \in \mathsf{Rel}_L$ in the 3rd round from $\mathcal{A}^{\mathsf{swi}}$ is the same in both $\mathcal{H}_2(1^\lambda, z)$ and $\mathcal{H}_3(1^\lambda, z)$. For the same arguments also the extraction of the signatures holds with the same probability in both $\mathcal{H}_2(1^\lambda, z)$ and $\mathcal{H}_3(1^\lambda, z)$.

We now prove that $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z) \approx \mathsf{wimim}_{\mathcal{H}_3}(1^\lambda, z)$ by contradiction constructing an adversary $\mathcal{A}^{\mathsf{SHVZK}}$ for the adaptive Special HVZK of LS. This adversary $\mathcal{A}^{\mathsf{SHVZK}}$ uses the adversary $\mathcal{A}^{\mathsf{swi}}$ and the distinguisher $\mathcal{D}^{\mathsf{swi}}$ that distinguishes $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z)$ from



$\mathcal{C}^{\mathsf{TC}_\Sigma}$
$\mathcal{A}^{\mathsf{TC}_\Sigma}$
$\mathcal{A}^{\mathsf{swi}}$
$\mathsf{vk}, \rho, \mathsf{ls}^2$
$\mathtt{com}$
$\mathtt{tk}$
$\mathsf{ls}^2, \mathtt{dec}, \ldots$
$\mathsf{vk}, \rho, \ldots$
$\mathtt{com}, \mathtt{msg}$
$\sigma, \ldots$
$\mathsf{ls}^2, \mathtt{dec}, \ldots$

Figure 5.8: During the reduction to the adaptive trapdoorness of TC_Σ could be required to play the left session multiple times because of the rewinds made in a $\mathsf{AllRewind}$ right session. In general this happen when it is required to recompute the second round of the left session, due to a new first round received by $\mathcal{A}^{\mathsf{swi}}$ (in the left session). In the figure it is also showed that, for every new interaction that $\mathcal{A}^{\mathsf{TC}_\Sigma}$ starts against $\mathcal{C}^{\mathsf{TC}_\Sigma}$, a new extraction of the trapdoor information $\mathtt{tk}$ is required in order to complete the reduction. We observe that the extraction of $\mathtt{tk}$ does not disturb the extraction made on the right session.

$\mathsf{wimim}_{\mathcal{H}_3}(1^\lambda, z)$ that exist by contradiction. Let $\mathcal{C}^{\mathsf{SHVZK}}$ be the challenger of adaptive Special HVZK. The reduction works as follows.

1. $\mathcal{A}^{\mathsf{SHVZK}}$ runs $\mathcal{A}^{\mathsf{swi}}$ (both on the left and on the right sessions) according to both $\mathcal{H}_2(1^\lambda, z)$ and $\mathcal{H}_3(1^\lambda, z)$ until the 3rd round of the left session is received from $\mathcal{A}^{\mathsf{swi}}$.
2. $\mathcal{A}^{\mathsf{SHVZK}}$ extracts from the left session two valid signatures for two different messages (as described before), and then sends to the challenger of Special HVZK $\mathcal{C}^{\mathsf{SHVZK}}$ the statement x the witness w (provided in the third round of the left session by $\mathcal{A}^{\mathsf{swi}}$) and the pair $(\mathsf{ls}^1, \mathsf{ls}^3)$ received in the 1st and 3rd round from $\mathcal{A}^{\mathsf{swi}}$.
3. $\mathcal{A}^{\mathsf{SHVZK}}$, upon receiving the messages $(\mathsf{ls}^2, \mathsf{ls}^4)$ from the $\mathcal{C}^{\mathsf{SHVZK}}$ uses them to compute and send the last round of SWI.
4. $\mathcal{A}^{\mathsf{SHVZK}}$ extracts from the $\mathsf{poly}(\lambda)$ right sessions the witnesses for the statements proved by $\mathcal{A}^{\mathsf{swi}}$ as follows.

   For the $\mathsf{NoRewind}$ and $\mathsf{AllRewind}$ right sessions the extractor procedure is the same as described before. That is, we use the PoK extractor of LS to extract the witnesses for the statements proved in these sessions. Note that the extractor of LS succeeds with non-negligible probability due to the adaptive-input AoK property of LS and the adaptive Special HVZK of LS. In more details if the extractor fails with non-negligible probability then the event NoExt holds with non-negligible probability, and it is possible to show an adversary $\mathcal{A}^{\mathsf{SHVZK}}$ that breaks the adaptive Special HVZK of LS. Indeed, if NoExt holds with non-negligible probability, then $\mathcal{A}^{\mathsf{SHVZK}}$ has a non-negligible advantage since we are proved earlier that NoExt holds with negligible probability when the values $\mathsf{ls}^2, \mathsf{ls}^4$ are computed using the honest prover procedure $\mathcal{P}$.

   In addition, for the $\mathsf{AllRewind}$ right sessions $\mathcal{A}^{\mathsf{SHVZK}}$ interacts against $\mathcal{A}^{\mathsf{swi}}$ executing the above steps, starting from step 1, every time that a first round is received in the left session, as showed in Figure 5.9. We remark that this is possible because we are considering a challenger for adaptive Special HVZK.

For the $\mathsf{3Rewind}$ sessions we need to use a different approach, due to fact that we cannot use reuse ls^2 in order to compute an accepting transcript for LS $(\mathsf{ls}^1, \mathsf{ls}^2, \mathsf{ls}^{3\prime}, \mathsf{ls}^{4\prime})$ with $\mathsf{ls}^3 \neq \mathsf{ls}^{3\prime}$. Therefore the extraction procedure of the witness for the statement $\tilde{x}_j$ proved by $\mathcal{A}^{\mathsf{swi}}$ in the j-th right session (with $j = 1, \dots, \mathsf{poly}(\lambda)$) works as following. The 4th round of the left session is played again by decommitting $(\rho, \mathtt{com})$ to a different value $\mathsf{ls}^{2\prime}$ generated by running $\mathcal{P}$ on input 1^λ, ℓ and ls^1. We remark that it is possible to open $(\rho, \mathtt{com})$ to a different value because we are using TFake_Σ to compute both commitment and decommitment informations. In order to ends up in obtaining that session j is completed successfully, multiple rewinds trying with different polynomial values for $\mathsf{ls}^{2\prime}$ could be required, until a successful $\mathsf{ls}^{2\prime}$ is found. Let $\tilde{x}_j^*$ be the statement proved by $\mathcal{A}^{\mathsf{swi}}$ in the j-th right session when a successful $\mathsf{ls}^{2\prime}$ is found. $\mathcal{A}^{\mathsf{SHVZK}}$ runs the extractor of LS in the j-th right session in order to extract the witness for the theorem $\tilde{x}_j^*$ (the same argument discussed above need to be used here in order to ensure that the extractor of LS succeeds with non-negligible probability). Observe that when it is required to recompute the 4th round during the extraction procedure, now it is possible to reuse $(\mathtt{dec}, \mathsf{ls}^{2\prime})$ in order to compute an accepting transcript for LS $(\mathsf{ls}^1, \mathsf{ls}^{2\prime}, \mathsf{ls}^{3\prime}, \mathsf{ls}^{4\prime})$ by running $\mathcal{P}$, for every $\mathsf{ls}^{3\prime}$ received by $\mathcal{A}^{\mathsf{swi}}$. If the extraction is successful then $\mathcal{A}^{\mathsf{SHVZK}}$ has obtained a witness for $\tilde{x}_j^*$. The crucial observation is that $\tilde{x}_j^*$ has the same witness of $\tilde{x}_j$. Indeed, in the security proof of this lemma, we are assuming the $\mathcal{A}^{\mathsf{swi}}$ can choose the statement to be proved in the last round of every right session, but the the family of statements $\mathcal{X}_w$ has to be fixed in the second round of every right sessions.

5. From what we argue above the extraction of the witnesses succeeds with non-negligible probability, and thus $\mathcal{A}^{\mathsf{SHVZK}}$ runs the distinguisher $\mathcal{D}^{\mathsf{swi}}$ on input the first γ-bits of each extracted witnesses to distinguish $\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z)$ from $\mathsf{wimim}_{\mathcal{H}_3}(1^\lambda, z)$ in order to break the Special HVZK of LS.

The proof ends by observing that the following holds:

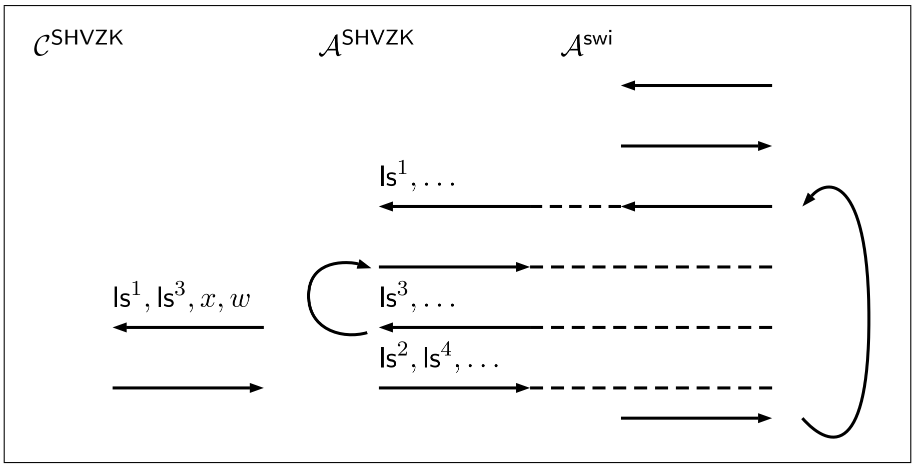
$$\mathsf{wimim}_{\mathsf{SWI}}(1^\lambda, z) = \mathsf{wimim}_{\mathcal{H}_1}(1^\lambda, z) \approx \mathsf{wimim}_{\mathcal{H}_1^{\mathsf{rew}}}(1^\lambda, z) \approx$$
$$\mathsf{wimim}_{\mathcal{H}_2}(1^\lambda, z) \approx \mathsf{wimim}_{\mathcal{H}_3}(1^\lambda, z) = \mathsf{sim}_{\mathsf{SWI}}^{\mathcal{S}^{\mathsf{swi}}}(1^\lambda, z).$$

5.6.2 Formal Proof of Th. 6

In this section we give more details on the hybrid experiments and on the security proof.

– The 1st experiment that we consider is $\mathcal{H}_1^m(z)$. In this, in the left session $\mathsf{NM4Sen}$ commits to m, while in the i-th right session, for $i = 1, \dots, \mathsf{poly}(\lambda)$, $\mathsf{NM4Rec}_i$ interacts with $\mathcal{A}^{\mathsf{NM4Com}}$. For ease of exposition we consider that the messages of SWI are organized as a query. That is, a MiM adversary of an execution of SWI expects to receive a query query that contains both the queries of the left and the right sessions. Following [GK96a, HRVW09], we make the following assumptions about the queries:

  – the same query is never asked twice.
  – a query is a partial transcript $(b_1, a_1, \dots, b_i)$ of the protocol. Moreover, whenever such query is made, all proper prefixes of this query were previously queried, namely all sequences of the form $(b_1, a_1, \dots, b_j)$ for $j < i$.



$\mathcal{C}^{\mathsf{SHVZK}}$
$\mathcal{A}^{\mathsf{SHVZK}}$
$\mathcal{A}^{\mathsf{swi}}$
$\mathsf{ls}^1, \ldots$
$\mathsf{ls}^1, \mathsf{ls}^3, x, w$
$\mathsf{ls}^3, \ldots$
$\mathsf{ls}^2, \mathsf{ls}^4, \ldots$

Figure 5.9: During the reduction to the adaptive Special HVZK of LS could be required to play the left session multiple times because of the rewinds made in the right session. In general this happen when it is required to recompute the 4th round of the left session, due to a new first round received by $\mathcal{A}^{\mathsf{swi}}$ (in the left session). In the figure it is also showed that, for every new interaction that $\mathcal{A}^{\mathsf{SHVZK}}$ starts against $\mathcal{C}^{\mathsf{SHVZK}}$, a new extraction of the trapdoor information `tk` is required in order to complete the reduction. We observe that the extraction of `tk` does not disturb the extraction made on the right session.

Therefore we can assume that a query has the following form $\mathsf{query} = (\mathsf{prefix}, \mathsf{q})$ where prefix are the messages of query previously queried, and q is a new message. The first message of a query is q^s and is the message sent to initiate new execution of SWI. Therefore when a query is composed only of the starting message $\mathsf{query} = (\mathsf{q}^s)$ the experiment starts $\mathcal{A}^{\mathsf{NM4Com}}$.

More formally the experiment $\mathcal{H}_1^m(z)$ is defined in the following way. The possible values of q w.r.t. the messages of $\mathcal{P}^{\mathsf{swi}}$ are the following.

- Messages $\mathsf{q}_2^{\mathsf{L}}$ is the message sent at the 2nd round of SWI protocol in a left session.
- Messages $\mathsf{q}_4^{\mathsf{L}}$ is the message sent at the 4th round of SWI protocol in a left session.

The possible values of q w.r.t. the messages of $\mathcal{V}_i^{\mathsf{swi}}$, with $i = 1, \ldots, \mathsf{poly}(\lambda)$, are the following.

- Messages $\mathsf{q}_1^{\mathsf{R},\mathsf{i}}$ is the message sent at the 1st round of SWI protocol in a i-th right session.
- Messages $\mathsf{q}_3^{\mathsf{R},\mathsf{i}}$ is the message sent at the 3rd round of SWI protocol in a i-th right session.

Let table be a lookup table, in which each row contains a query query and the messages (and the randomness used to compute these messages) of Π_{wom} computed when query was processed. We refer to that messages as τ_{wom}. When a query $\mathsf{query} = (\mathsf{prefix}, \mathsf{q})$ is received the experiment looks up in table for the query prefix in order to recover τ_{wom}. Then, the experiment reconstructs the partial transcript τ_{NM4Com} of Π_{NM4Com} computed according to the messages of prefix. We assume without loss of generality that $\mathcal{A}^{\mathsf{NM4Com}}$ is deterministic, therefore the experiment reconstructs τ_{NM4Com} interacting with $\mathcal{A}^{\mathsf{NM4Com}}$ using the messages of τ_{wom} contained in prefix. After that, the new message q is processed and depending on the value of q, the experiment acts in different way. More precisely, if q is a message of $\mathcal{P}^{\mathsf{swi}}$, the experiment interacts with $\mathcal{A}^{\mathsf{NM4Com}}$ acting as NM4Sen would do for the messages of Π_{wom} forwarding the messages contained in query in order to act as $\mathcal{P}^{\mathsf{swi}}$. Similarly, if q is a message of the right session, let us say, a message from $\mathcal{V}_i^{\mathsf{swi}}$, the experiment interacts with $\mathcal{A}^{\mathsf{NM4Com}}$ acting as $\mathsf{NM4Rec}_i$ would do for the messages of Π_{wom} and forwarding to $\mathcal{A}^{\mathsf{NM4Com}}$ q. When the execution against $\mathcal{A}^{\mathsf{NM4Com}}$ finishes a new entry in table is created in order to memorize the received query and τ'_{wom}, where τ'_{wom} contains the messages (and the randomness used to compute these messages) of Π_{wom} obtained when query is processed. Then the experiment, collects the messages w.r.t. SWI received by $\mathcal{A}^{\mathsf{NM4Com}}$ in an answer answer, that is the answer to the asked query. More formally, the experiment works as following.

Upon receiving $\mathsf{query} = (\mathsf{prefix}, \mathsf{q})$, If $\mathsf{query} = \mathsf{q}^s$ then start $\mathcal{A}^{\mathsf{NM4Com}}$, otherwise execute the following steps.

1. Look up in table for a query that correspond to prefix and recover τ_{wom}.
2. In the left session interact with $\mathcal{A}^{\mathsf{NM4Com}}$ forwarding the messages contained in prefix w.r.t. SWI in order to act as $\mathcal{P}^{\mathsf{swi}}$ would do. Use the messages of prefix of τ_{wom} in order to act as $\mathsf{NM4Sen}$ would do.
3. In the i-th right session, for $i = 1, \dots, \mathsf{poly}(\lambda)$, forwarding the messages contained in prefix w.r.t. SWI in order to act as $\mathcal{V}_i^{\mathsf{swi}}$ would do. Furthermore, interact with $\mathcal{A}^{\mathsf{NM4Com}}$ using the messages of prefix of τ_{wom} in order to act as $\mathsf{NM4Rec}_i$.
4. Set $\tau'_{\mathsf{wom}} = \tau_{\mathsf{wom}}$ and process q computing the following steps.

   Left session.
   - If $\mathsf{q} = \mathsf{q}_2^{\mathsf{L}}$ then computes the following steps. Upon receiving $\mathsf{wom}^1, \mathsf{swi}^1$ from $\mathcal{A}^{\mathsf{NM4Com}}$, run $\mathsf{Sen}_{\mathsf{wom}}$ on input 1^λ, $\mathtt{id}$, wom^1 and m thus obtaining the 2nd round wom^2 of Π_{wom} and $\mathtt{dec}_{\mathsf{wom}}$ s.t. $(m, \mathtt{dec}_{\mathsf{wom}})$ constitutes the decommitment information. Add $(\mathsf{wom}^2, m, \mathtt{dec}_{\mathsf{wom}})$ (according to the randomness used to compute it) to τ'_{wom}. Recover swi^2 from $\mathsf{q}_2^{\mathsf{L}}$ and send $(\mathsf{wom}^2, \mathsf{swi}^2)$ to $\mathcal{A}^{\mathsf{NM4Com}}$.
   - if $\mathsf{q} = \mathsf{q}_4^{\mathsf{L}}$ then computes the following steps. Upon receiving $\mathsf{wom}^3, \mathsf{swi}^3$ from $\mathcal{A}^{\mathsf{NM4Com}}$ run $\mathsf{Sen}_{\mathsf{wom}}$ on input wom^3 thus obtaining the 4th round wom^4 of Π_{wom}. Add wom^4 (according to the randomness used to compute it) to τ'_{wom} and send $(\mathsf{wom}^4, \mathsf{swi}^4)$ to $\mathcal{A}^{\mathsf{NM4Com}}$.

   i-th right session.
   - if $\mathsf{q} = \mathsf{q}_1^{\mathsf{R,i}}$ computes the following steps. Run $\mathsf{Rec}_{\mathsf{wom}}$ on input $1^\lambda, \tilde{\mathtt{id}}_i$ thus obtaining the 1st round $\tilde{\mathsf{wom}}_i^1$ of Π_{wom}. Update τ'_{wom} with $\tilde{\mathsf{wom}}_i^1$ and the randomness used to compute it. Recover $\tilde{\mathsf{swi}}_i^1$ from $\mathsf{q}_1^{\mathsf{R,i}}$ and send $(\tilde{\mathsf{wom}}_i^1, \tilde{\mathsf{swi}}_i^1)$ to $\mathcal{A}^{\mathsf{NM4Com}}$.
   - If $\mathsf{q} = \mathsf{q}_3^{\mathsf{R,i}}$ computes the following steps. Upon receiving $(\tilde{\mathsf{wom}}_i^2, \tilde{\mathsf{swi}}_i^2)$ from $\mathcal{A}^{\mathsf{NM4Com}}$. Run $\mathsf{Rec}_{\mathsf{wom}}$ on input $\tilde{\mathsf{wom}}_i^2$ thus obtaining the 3rd round $\tilde{\mathsf{wom}}_i^3$ of Π_{wom}. Update τ'_{wom} with $\tilde{\mathsf{wom}}_i^3$ and the randomness used to compute it. Recover $\tilde{\mathsf{swi}}_i^3$ from $\mathsf{q}_3^{\mathsf{R,i}}$ and send $(\tilde{\mathsf{wom}}_i^3, \tilde{\mathsf{swi}}_i^3)$ to $\mathcal{A}^{\mathsf{NM4Com}}$.
5. Construct the answer answer for the query query as following.

   5.1. In the left session if $(\mathsf{swi}^1, \mathsf{wom}^1)$ is received add swi^1 to answer. Otherwise if $(\mathsf{swi}^3, \mathsf{wom}^3)$ is received recover $\mathsf{wom}^1, \mathsf{wom}^2, m, \mathtt{dec}_{\mathsf{wom}}$ from τ'_{wom},
   set $x = (\mathsf{wom}^1, \mathsf{wom}^2, \mathsf{wom}^3, \mathsf{wom}^4, \mathtt{id})$ and $w = (m, \mathtt{dec}_{\mathsf{wom}})$, and add $(x, w, \mathsf{swi}^1, \mathsf{swi}^3)$ to answer.

   5.2. For the i-th right session add to answer the messages of SWI received from $\mathcal{A}^{\mathsf{NM4Com}}$ in i-th right session, for $i = 1, \dots, \mathsf{poly}(\lambda)$.
6. Memorize query and τ'_{wom} in a new row of table and return answer.

Clearly we have that $\mathsf{mim}_{\Pi_{\mathsf{NM4Com}}}^{\mathcal{A}_{\mathsf{NMCom}}, m}(z) = \mathsf{mim}_{\mathcal{H}_1^m}^{\mathcal{A}_{\mathsf{NMCom}}, m}(z)$. We prove the following claim.

Claim 4. *Let $\bar{p}_i$ be the probability that for $i = 1, \dots, \mathsf{poly}(\lambda)$ in the i-th right sessions of $\mathcal{H}_1^m(z)$ $\mathcal{A}^{\mathsf{NM4Com}}$ successfully commits to a message $\tilde{m}_i = \perp$, then $\bar{p}_i < \nu(\lambda)$ for some negligible function ν.*

Proof. We prove this claim by contradiction: we assume that there exists a right session i where $\mathcal{A}^{\mathsf{NM4Com}}$ commits to $\perp$, then the statement proved by SWI in the i-th right session is false, this contradict the adaptive-input AoK property of SWI. In more details, we can use the extractor of SWI in order to extract the witness $\tilde{w}_i$ s.t. $(\tilde{x}_i, \tilde{w}_i) \in \mathsf{Rel_L}$, where $\tilde{x}_i$ is the statement proved by $\mathcal{A}^{\mathsf{NM4Com}}$ in i-th right session. Since the extractor of SWI succeeds with non-negligible probability we can recover with non-negligible probability the witness $\tilde{w}_i$ that correspond to the decommitment informations of the well-formed commitment computed by $\mathcal{A}^{\mathsf{NM4Com}}$ in i-th right session. This implies that $\mathcal{A}^{\mathsf{NM4Com}}$ commits to a message $\tilde{m}_i = \perp$ in i-th right session with negligible probability. □

– We consider the experiment $\mathcal{H}_1^0(z)$ that corresponds to $\mathcal{H}_1^m(z)$ with the only difference that the message committed using Π_{wom} is 0^λ instead of m. We observe that $\mathsf{mim}_{\mathcal{H}_1^0}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z) = \mathsf{sim}_{\Pi_{\mathsf{NM4Com}}}^{\mathsf{Sim}^{\mathsf{NM4Com}}}(1^\lambda, z)$ and we prove the following claim.

Claim 5. *Let $\bar{p}_i$ be the probability that for $i = 1, \dots, \mathsf{poly}(\lambda)$ in the i-th right sessions of $\mathcal{H}_1^0(z)$ $\mathcal{A}^{\mathsf{NM4Com}}$ successfully commits to a message $\tilde{m}_i = \perp$, then $\bar{p}_i < \nu(\lambda)$ for some negligible function ν.*

The security proof of this claim follows strictly the one of Claim 4.

– We now consider the 2nd hybrid experiment $\mathcal{H}_2^m(z)$ that differs from $\mathcal{H}_1^m(z)$ in the way the transcript for SWI is computed. More precisely, in this case the transcript for SWI is computed by using the simulator $\mathcal{S}^{\mathsf{swi}}$[15] of SWI. In the hybrid experiment $\mathcal{H}_2^m(z)$ we process the queries query made by $\mathcal{S}^{\mathsf{swi}}$ in the same way as we described in the hybrid $\mathcal{H}_1^m(z)$. We prove the following claim.

Claim 6. *For all $m \in \{0,1\}^{\mathsf{poly}(\lambda)}$ it holds that $\mathsf{mim}_{\mathcal{H}_1^m}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z) \approx \mathsf{mim}_{\mathcal{H}_2^m}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z)$.*

Suppose by contradiction that there exist adversary $\mathcal{A}^{\mathsf{NM4Com}}$ and a distinguisher $\mathcal{D}^{\mathsf{NM4Com}}$ that can tell apart such two distributions. We can use this adversary and the associated distinguisher to construct a MiM adversary $\mathcal{A}^{\mathsf{swi}}$ and a distinguisher $\mathcal{D}^{\mathsf{swi}}$ that break the adaptive-input SimWI property of SWI. Observe that this reduction is possible due to the adaptiveness of SWI; indeed, in every right sessions the statement proved by $\mathcal{A}^{\mathsf{NM4Com}}$ is fully defined only in the last round. We also recall that even if the statements to be proved is specified in the last round, the family of theorems $\mathcal{X}_w$ has to be fixed at second round. Since Π_{wom} fixes the decommitment informations in the second round, then also $\mathcal{X}_w$ is implicitly fixed at second round.

Let $\mathcal{C}^{\mathsf{swi}}$ be the challenger of SimWI. $\mathcal{A}^{\mathsf{swi}}$ interacts with $\mathcal{A}^{\mathsf{NM4Com}}$ forwarding the messages of SWI that receive from $\mathcal{C}^{\mathsf{swi}}$ and computing the messages of Π_{wom} on his own. In more details $\mathcal{A}^{\mathsf{swi}}$ acts against $\mathcal{C}^{\mathsf{swi}}$ processing his queries in the same way that is described in both the experiments $\mathcal{H}_1^m(z)$ and $\mathcal{H}_2^m(z)$. Let's now describe how $\mathcal{D}^{\mathsf{swi}}$ works. $\mathcal{D}^{\mathsf{swi}}$, takes as input the view $\mathcal{A}^{\mathsf{swi}}$ the first γ-bits of the witness $\tilde{w}_1, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}$, namely $\tilde{w}_1^\gamma, \dots, \tilde{w}_{\mathsf{poly}(\lambda)}^\gamma$,

[15]Following [GK96a, HRVW09] we assume that if $(b_1, a_1, \dots, b_m, a_m)$ is the transcript that appears in the final output of $\mathcal{S}^{\mathsf{swi}}$, then $\mathcal{S}^{\mathsf{swi}}$ has queried $\mathcal{A}^{\mathsf{swi}}$ on $(b_1, a_1, \dots, b_m, a_m)$.

associated to the statement proved by $\mathcal{A}^{\mathsf{swi}}$ and the bits $b_1, \dots, b_{\mathsf{poly}(\lambda)}$ s.t. $w_i^\gamma = \bot$ iff $b_i = 0$ with $i = 1, \dots, \mathsf{poly}(\lambda)$.

Every $\tilde{w}_i^\gamma \neq \bot$ corresponds to the message $\tilde{m}_i$ of the commitment computed using Π_{wom} in i-th right session, for $i = 1, \dots, \mathsf{poly}(\lambda)$. Therefore, $\mathcal{A}^{\mathsf{swi}}$ recovers the messages $\tilde{m}_1, \dots, \tilde{m}_{\mathsf{poly}(\lambda)}$ committed in the right sessions setting $\tilde{m}_i = \bot$ if $\tilde{w}_i^\gamma = \bot$. Finally, $\mathcal{D}^{\mathsf{swi}}$ reconstructs the view of $\mathcal{A}^{\mathsf{NM4Com}}$ (by using the randomness of $\mathcal{A}^{\mathsf{swi}}$ contained in $\mathcal{A}^{\mathsf{swi}}$'s view) and gives it and the messages $\tilde{m}_1, \dots, \tilde{m}_{\mathsf{poly}(\lambda)}$ as inputs of $\mathcal{D}^{\mathsf{NM4Com}}$ giving in output what $\mathcal{D}^{\mathsf{NM4Com}}$ outputs. Since by contradiction $\mathcal{D}^{\mathsf{NM4Com}}$ distinguishes between $\mathsf{mim}_{\mathcal{H}_1^m}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z)$ and $\mathsf{mim}_{\mathcal{H}_2^m}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z)$ we have that $\mathcal{D}^{\mathsf{swi}}$ tells apart whether $\mathcal{A}^{\mathsf{swi}}$ interacts with the simulator of SWI or not. The proof ends with the observation that if $\mathcal{C}^{\mathsf{swi}}$ computes the transcript of SWI as the simulator for SWI does, then $\mathcal{A}^{\mathsf{NM4Com}}$ acts as in $\mathcal{H}_2^m(z)$, otherwise he acts as in $\mathcal{H}_1^m(z)$.

– We consider the experiment $\mathcal{H}_2^0(z)$ that corresponds to $\mathcal{H}_2^m(z)$ with the only difference that $\mathsf{NM4Sen}$ commits using Π_{wom} to a message 0^λ instead of m. We prove the following claims.

Claim 7. *For all $m \in \{0,1\}^{\mathsf{poly}(\lambda)}$ it holds that $\mathsf{mim}_{\mathcal{H}_1^0}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z) \approx \mathsf{mim}_{\mathcal{H}_2^0}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z)$.*

The security proof of this claim follows strictly the one of Claim 6.

Claim 8. *For all $m \in \{0,1\}^{\mathsf{poly}(\lambda)}$ it holds that $\mathsf{mim}_{\mathcal{H}_2^m}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z) \approx \mathsf{mim}_{\mathcal{H}_2^0}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z)$.*

The proof follows from the weak concurrent non-malleability property of Π_{wom}. Indeed observe that here it suffices to rely on the *weak* concurrent NM commitment because we are guaranteed (from the previous arguments) that whenever $\mathcal{A}^{\mathsf{NM4Com}}$ completes a commitment in the i-th right session, with $i = 1, \dots, \mathsf{poly}(\lambda)$, the corresponding message committed through Π_{wom} is different from $\bot$ with overwhelming probability in both $\mathcal{H}_2^m(z)$ and $\mathcal{H}_2^0(z)$. To prove the indistinguishability between those two hybrids experiments we proceed by contradiction constructing an adversary $\mathcal{A}^{\mathsf{wom}}$ that breaks the *weak* concurrent non-malleability property of Π_{wom}.

Let $\mathcal{C}^{\mathsf{wom}}$ be the challenger of the weak concurrent NM commitment and let $m_1, \dots, m_{\mathsf{poly}}(\lambda)$ with $m, 0^\lambda$ be the challenge messages. $\mathcal{A}^{\mathsf{wom}}$ processes the queries sent by $\mathcal{S}^{\mathsf{swi}}$ as described in both $\mathcal{H}_2^m(z)$ and $\mathcal{H}_2^0(z)$, with the following difference. In the left session, for the messages of Π_{wom}, $\mathcal{A}^{\mathsf{wom}}$ acts as a proxy between $\mathcal{C}^{\mathsf{wom}}$ and $\mathcal{A}^{\mathsf{NM4Com}}$. The same happen in the right sessions, where $\mathcal{A}^{\mathsf{wom}}$ acts as a proxy between $\mathcal{A}^{\mathsf{NM4Com}}$ ad the receiver $\mathsf{Rec}_{\mathsf{wom},i}$ for $i = 1, \dots, \mathsf{poly}(\lambda)$. More precisely $\mathcal{A}^{\mathsf{wom}}$ manages the queries made by $\mathcal{S}^{\mathsf{swi}}$ as describe in both $\mathcal{H}_2^m(z)$ and $\mathcal{H}_2^0(z)$, but instead of internally run $\mathsf{Sen}_{\mathsf{wom}}$ and $\mathsf{Rec}_{\mathsf{wom}}$, in order to compute the messages of Π_{wom}, $\mathcal{C}^{\mathsf{wom}}$ and the external receivers $\mathsf{Rec}_{\mathsf{wom},1} \dots, \mathsf{Rec}_{\mathsf{wom},\mathsf{poly}(\lambda)}$ involved in the reduction are used. In this setting there are two caveats that $\mathcal{A}^{\mathsf{wom}}$ needs to address in order to successfully complete the reduction.

The first one is that $\mathcal{S}^{\mathsf{swi}}$ could rewind the entire left session during the reduction, therefore requiring to compute a new commitment for the protocol Π_{wom}. Since we are assuming that Π_{wom} is concurrent, $\mathcal{A}^{\mathsf{wom}}$ can request to receive multiple commitments for the same message m during the reduction in order to interact against $\mathcal{A}^{\mathsf{NM4Com}}$ in such a scenario. The second one is that the rewinds made by $\mathcal{S}^{\mathsf{swi}}$ in the left session, could rewind the right sessions. Without loss of generality we suppose that right session affected by the rewinds made in the left session is the j-th, with $j \in \{1, \dots, \mathsf{poly}(\lambda)\}$. If the entire right session is rewound, then $\mathcal{A}^{\mathsf{wom}}$ simply asks to start another execution with a new receiver $\mathsf{Rec}_{\mathsf{wom}}{}'_{,j}$ of Π_{wom}. Could also happen that $\mathsf{Rec}_{\mathsf{wom},j}$ is rewound from the third to the second round, that is, a new $\mathsf{wom}_j^{2\prime}$ is sent by $\mathcal{A}^{\mathsf{NM4Com}}$ in the right session that we are considering. In that case $\mathcal{A}^{\mathsf{wom}}$ does not forward $\mathsf{wom}_j^{2\prime}$ to $\mathsf{Rec}_{\mathsf{wom},j}$, but compute a random $\mathsf{wom}_j^{3\prime}$ and use it to reply to $\mathcal{A}^{\mathsf{NM4Com}}$. We observe that it is possible to compute $\mathsf{wom}_j^{3\prime}$ without knowing the randomness used by $\mathsf{Rec}_{\mathsf{wom},j}$ so far because Π_{wom} is public coin.

Now we are ready to show how $\mathcal{D}^{\mathsf{wom}}$ works. $\mathcal{D}^{\mathsf{wom}}$, on input the messages $\tilde{m}_1, \dots, \tilde{m}_{\mathsf{poly}(n)}$ committed by $\mathcal{A}^{\mathsf{NM4Com}}$ and the view of $\mathcal{A}^{\mathsf{wom}}$, reconstructs the view that $\mathcal{A}^{\mathsf{NM4Com}}$ has when he played in the execution that generated the transcript of SWI given in output by $\mathcal{S}^{\mathsf{swi}}$. Then, $\mathcal{D}^{\mathsf{wom}}$ uses the view of $\mathcal{A}^{\mathsf{NM4Com}}$ along with $m, 0^{\lambda}$ and the messages $\tilde{m}_1, \dots, \tilde{m}_{\mathsf{poly}(n)}$ as inputs of $\mathcal{D}^{\mathsf{NM4Com}}$ giving in output what $\mathcal{D}^{\mathsf{NM4Com}}$ outputs. Since by contradiction $\mathcal{D}^{\mathsf{NM4Com}}$ distinguishes between $\mathsf{mim}_{\mathcal{H}_2^m}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z)$ and $\mathsf{mim}_{\mathcal{H}_2^0}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z)$ also $\mathcal{D}^{\mathsf{wom}}$ can tell apart which messages was committed by $\mathcal{C}^{\mathsf{wom}}$. The proof ends with the observation that if $\mathcal{C}^{\mathsf{wom}}$ commits to m $\mathcal{A}^{\mathsf{NM4Com}}$ acts as in $\mathcal{H}_2^m(z)$, otherwise he acts as in $\mathcal{H}_2^0(z)$.

The proof ends by observing that for all $m \in \{0,1\}^{\mathsf{poly}(\lambda)}$ the following holds:

$$\mathsf{mim}_{\Pi_{\mathsf{NM4Com}}}^{\mathcal{A}_{\mathsf{NMCom}},m}(z) = \mathsf{mim}_{\mathcal{H}_1^m}^{\mathcal{A}_{\mathsf{NMCom}},m}(z) \approx \mathsf{mim}_{\mathcal{H}_2^m}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z) \approx$$
$$\approx \mathsf{mim}_{\mathcal{H}_2^0}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z) \approx \mathsf{mim}_{\mathcal{H}_1^0}^{\mathcal{A}^{\mathsf{NM4Com}},m}(z) = \mathsf{sim}_{\Pi_{\mathsf{NM4Com}}}^{\mathsf{Sim}^{\mathsf{NM4Com}}}(1^{\lambda}, z).$$

Part III

Efficient Proof Systems

Chapter 6

Delayed-Input Witness Indistinguishable Proofs of Knowledge

6.1 Introduction

Proofs of knowledge (PoKs) offer a very strong security guarantee to a verifier of an interactive proof. When this property is coupled with an adequate security notion for the prover, such as honest-verifier zero knowledge (HVZK), witness indistinguishability (WI) or zero knowledge (ZK), the resulting proofs can be used as building blocks in essentially every protocol for secure computation. As such, the degree of security and efficiency achieved by the underlying PoKs, directly and dramatically, impacts on the security and efficiency of the larger protocol. For instance, the existence of very efficient WI PoKs for specific languages such as Discrete Log and DDH has been instrumental for constructing efficient maliciously secure two-party computation (see [HL10] and reference within). Round-efficient protocols [Pas03b, KO04] require security notions, both with respect to a malicious prover (soundness) and with respect to a malicious verifier (WI), that hold even in presence of *adaptive-input* selection.

Proofs of partial knowledge. In [CDS94], Cramer et al. showed that one can use a specific type of three-round public-coin PoK (called a Σ-protocol) to construct an *efficient* PoK for a compound statement. More precisely, the compound statement consists of n instances, and the goal is to prove knowledge of a witness for at least k of the n instances. As such, these proofs are named "(k, n)*-proofs of partial knowledge*" in [CDS94]. The transform of [CDS94] cleverly combines n parallel executions Σ-protocols in an efficient 3-round public-coin perfect WI (k, n)-proof of partial knowledge. We stress that, even though the starting Σ-protocol only enjoyed security with respect to passive verifiers (in the form of HVZK), the resulting proof of partial knowledge offers some level of security even against active verifiers (in the form of perfect WI). A similar result was given in [DSDCPY94] for perfect ZK.

Note that, if efficiency is not a concern, three-round public coin proofs of partial knowledge were already possible (with *computational* WI, though) thanks to the general construction of Lapidot and Shamir (LaSh) [LS90][1]. Proving compound statements via LaSh however requires expensive $\mathcal{NP}$ reductions. On the other hand, LaSh PoKs provide a stronger security guarantee: honest players use the instances specified in the statements only in the last round, and security holds even if the adversarial verifier (resp., prover) chooses the instances adaptively after having seen the first (resp., second) round. Specifically, LaSh PoKs are *adaptive-input* WI proofs of partial knowledge for all $\mathcal{NP}$. When used as part of a larger protocol, the ability to start the

[1]See [OV12] for a detailed description of [LS90].

proof of partial knowledge before the input is actually produced by the larger protocol saves at least one round of communication.

The construction shown in [CDS94], instead, although efficient, does not provide any form of adaptivity, as all the n instances must be fully specified before the protocols starts. As a consequence, the improved efficiency of [CDS94] must be paid for by the additional rounds needed by the larger protocol that uses the PoK as a building block.

A first step. A very recent work by Ciampi et al. [CPS$^+$16a] makes a first preliminary step towards closing the gap between [LS90] and [CDS94]. Ciampi et al. propose a different transform for WI proofs of partial knowledge that gives some adaptivity at the price of generality. Namely, their technique yields to a $(1, 2)$-proof of partial knowledge where the knowledge of one of the two instances can be postponed to the last round. In more details, they show a PoK for a statement "$x_0 \in L_0 \vee x_1 \in L_1$" in which x_0 and x_1 are not immediately needed (in contrast to [CDS94]). The honest prover needs x_0 to run the first round while x_1 is needed only in the 3rd round along with a witness for either one of x_0 and x_1. The verifier needs to see x_0 and x_1 only at the end, in order to accept/reject the proof. These PoKs are called delayed input in [CPS$^+$16a] as the need of the input is delayed to the very last round for the honest prover. For clarity, we stress that a delayed-input protocol is not necessarily secure against inputs that have been adaptively chosen. Indeed, the technique of [CPS$^+$16a] yields a proof of partial knowledge that is delayed input for one of the two instances, is adaptive-input WI but it is not adaptively secure against a malicious prover. The security achieved is sufficient for their target applications, though.

The open question and its importance. The above preliminary progress leaves open the following fascinating question: can we design an efficient transform that yields an adaptive-input WI (k, n)-proof of partial knowledge where *all* n instances are known only in the last round?

Previous efficient transforms require the a-priori knowledge of all instances or of one out of two instances, even if the corresponding languages admit efficient delayed-input Σ-protocols. For the sake of concreteness, consider the well known Σ-protocol Σ^{dl}, due to Schnorr [Sch89] for proving knowledge of discrete log. Schnorr's protocol is easily seen to be delayed-input as the prover needs not to know the instance $y = g^x$ in order to compute the first round. Now suppose one wants to prove knowledge of the discrete logarithm of at least one of $y_0 = g^{x_0}$ and $y_1 = g^{x_1}$. When we apply known transforms, the resulting protocol loses the delayed-input property. More specifically, both y_0 and y_1 are needed by [CDS94], and at least one of y_0 and y_1 is needed by [CPS$^+$16a].

6.1.1 Our Results

In this work we study the above open question and give various positive answers.

Σ-protocols and adaptive-input selection. We shed light on the relationship between delayed-input Σ-protocols (in which honest parties need not to know the input statement at the onset of the protocol) and adaptive-input Σ-protocols (that retain their security properties even if the input statement is adaptively chosen after seeing the first messages of the protocol). Recall that a Σ-protocol enjoys a special soundness property, which means that, given two accepting transcripts[2] for the same statement having the same first round, one can efficiently extract a witness for that statement.

[2]In the literature special soundness is often generalized to $\ell > 2$ accepting transcripts with the bound of ℓ being polynomial in the security parameter.

We show that delayed-input Σ-protocols are not necessarily adaptive-input sound; that is, they are not sound if the malicious prover can choose the statements adaptively. Indeed, in Section 6.3.1 we show how a malicious prover, based on the second round played by the verifier, can craft a false statement that will make the verifier accept and the extractor of special soundness fail. The extraction fails even when the statement is true. The attack applies to the most commonly used Σ-protocols, such as Schnorr's protocol for discrete logarithm, the protocol for Diffie-Hellman (DH) tuples and the protocol of [MP03] for proving knowledge of committed messages, and to all Σ-protocols in the well known class proposed by Cramer in [CD98] and Maurer in [Mau15].

The loss of soundness with respect to provers that adaptively choose their inputs was already noticed in [BPW12] for non-interactive zero-knowledge arguments obtained from Σ-protocols by means of the Fiat-Shamir transform [FS86]. Indeed there are in the literature some incorrect uses of the Fiat-Shamir transform in which an adversarial prover can first create a transcript and then can try to find an instance not in the language such that the transcript is accepting. In the random-oracle model the above issue can be addressed by using also the instance as input to the random oracle to generate the challenge. This fix is meaningless in the standard model that is the focus of our work.

We then analyze the transform of [CPS$^+$16a] that is delayed-input with respect to one instance only. We observe that when [CPS$^+$16a] combines protocols belonging to the class of [CD98, Mau15], it is not secure with respect to a malicious prover that is allowed to adaptively choose his input. Therefore the transform of [CPS$^+$16a] is not adaptive-input sound. We stress however, that in the applications targeted in [CPS$^+$16a] the input that is specified only in the last round is chosen by the verifier. As such, for their applications they do not need any form of adaptive-input soundness, but only adaptive-input witness-indistinguishability (which they achieve). Moreover, the special soundness of their transform preserves security w.r.t. adaptive-input selection. Summing up, [CPS$^+$16a] correctly defines and achieves delayed-input Σ-protocols and adaptive-input WI and uses it in the applications. However adaptive-input special soundness is not defined and not achieved in their work.

Adaptive-input special-sound Σ-protocols. In light of the above discussion, a natural question is whether we can upgrade the security of the class of Σ-protocols that are delayed input, but not adaptive-input sound.

Towards this, we first clarify the conceptual gap between adaptive-input selection and the adaptivity considered in [CPS$^+$16a] by formally defining adaptive-input special soundness. Then we show a compiler that takes as input any delayed-input Σ-protocol belonging to the class specified in [CD98, Mau15], and outputs a Σ-protocol that is adaptive-input sound; i.e., it is sound even when the malicious prover adaptively chooses his input in the last round.

The main idea behind this compiler is to force the prover to correctly send the first round of the Σ-protocol through another parallel run of the Σ-protocol. This allows for the extraction of any witness in the proof of knowledge. The compiler is shown in Section 6.3.2.

We also show (in Section 6.3.3) that nevertheless, [CPS$^+$16a]'s transform preserves the adaptivity of the Σ-protocols that are combined. Namely, when applied to Σ-protocols that are already adaptive-input special sound and WI, [CPS$^+$16a]'s transform outputs a $(1, 2)$-proof of partial knowledge that is an adaptive-input proof of knowledge as well.

Adaptive-input (k, n)-proofs of partial knowledge. The main contribution of this chapter is a new transform that yields the first efficient (k, n)-proofs of partial knowledge where *all* n instances can be specified in the last round.

Our new transform takes as input a delayed-input Σ-protocol for a relation Rel, and outputs a 3-round public-coin WI special-sound (k, n)-proof of partial knowledge for the relation $(\mathsf{Rel} \vee \cdots \vee \mathsf{Rel})$ where no instance is known at the beginning. The security of our transform is based on the DDH assumption. The WI property of the resulting protocol holds also with respect to adaptive-input selection, while the PoK property holds also in case of adaptive-input selection only if the underlying Σ-protocol is adaptive-input special sound.

We also show a transform that admits instances taken from different relations. Interestingly, this construction makes use as subprotocol of the first construction where instances are taken from the same relation.

6.1.2 Our Techniques

We provide a technique for composing a delayed-input Σ-protocol for a relation Rel in an delayed-input Σ-protocol for the (k, n)-proof of partial knowledge for relation $(\mathsf{Rel} \vee \ldots \vee \mathsf{Rel})$.

For better understanding our technique, it is instructive to see why the previous transformation [CDS94] (resp., [CPS$^+$16a]) requires that all n (resp., 1 out of 2) instances are specified before the protocol starts.

Limitations of previous transforms. Let Σ_{Rel} be a delayed-input Σ-protocol, and let $(\mathsf{Rel} \vee \ldots \vee \mathsf{Rel})$ be the relation for which we would like to have a (k, n)-proof of partial knowledge. The technique of [CDS94] works as follows. The prover P, on input the instances $(x_1 \in \mathsf{Rel} \vee \ldots \vee x_n \in \mathsf{Rel})$, runs protocols $\Sigma_{\mathsf{Rel}}, \ldots, \Sigma_{\mathsf{Rel}}$ in parallel. P gets only k witnesses for k different instances but it needs to somehow generate an accepting transcript for *all* instances. How to prove the remaining $n-k$ instances without having the witness? The idea of [CDS94] consists simply in letting the prover generate the $n-k$ transcripts (corresponding to the instances for which he did not get the witnesses) using the HVZK simulator Sim associated to the Σ-protocol. Additionally [CDS94] introduces a mechanism that allows the prover to control the value of exactly $(n-k)$ of the challenges played by V, so that the prover can force the transcripts computed by the simulator in $(n-k)$ positions.

So, why does the transform of [CDS94] need *all* instances to be known already in the 1st round? The answer is that P needs to run Sim already in the 1st round, and Sim expects the instance as input. Similar arguments apply for [CPS$^+$16a] as it requires that 1 instance out of 2 is known already in the 1st round.

The core idea of our technique. Previous transforms fail because the prover runs the HVZK simulator to compute the 1st round of some of the transcripts of Σ_{Rel}. Our core idea is to provide mechanisms allowing P to postpone the use of the simulator to the 3rd round. The main challenge is to implement mechanisms that are very efficient and preserve soundness and WI of the composed Σ-protocol. We stress that we want to solve the open problems in full, and thus none of the instances are known at the beginning of the protocol. To be more explicit, in the 1st round, the prover starts with the following statement $(? \in L_{\mathsf{Rel}} \vee \ldots \vee ? \in L_{\mathsf{Rel}})$.

Assume we have a (k, n)-equivocal commitment scheme that allows the prover to compute n commitments such that k of them are $\mathsf{binding}$ and the remaining $n-k$ are $\mathsf{equivocal}$, and the verifier cannot distinguish between the two types of commitment, where the k positions that are binding must be chosen already in the commitment phase (a similar tool was constructed in [ORS15]). With this gadget in hand, we can construct a delayed-input (k, n)-proof of partial knowledge $\Sigma^{\mathsf{OR}}_{k,n}$ as follows. Let (a, c, z) denote generically the 3 messages exchanged during the execution of a Σ-protocol Σ_{Rel}.

In the 1st round, P honestly computes a_i for the i-th execution of Σ_{Rel}. Here we are using the fact that Σ_{Rel} is delayed-input, and thus a_i can be computed without using the instance. Then he commits to $a_1, \ldots, a_n$ using the (k, n)-equivocal commitment scheme discussed above, where the k binding positions are randomly chosen. Thus, the 1st round of protocol $\Sigma^{\mathsf{OR}}_{k,n}$ consists of n commitments. In the 2nd round V simply sends a single challenge c according to Σ_{Rel}. In the 3rd round, P obtains the n instances $x_1, \ldots, x_n$ and k witnesses. At this point, for the instances x_i for which he did not receive the witness, he will use the HVZK simulator to compute an accepting transcript $(\tilde{a}_i, c, \tilde{z}_i)$ and then equivocate the $(n - k)$ equivocal commitments so that they decommit to the new generated $\tilde{a}_i$. For the k remaining instances he will honestly compute the 3rd round using the committed input a_i. Intuitively, soundness follows from the fact that k commitments are binding, and from the soundness of Σ_{Rel}. WI follows from the hiding of the equivocal commitment scheme and the HVZK property of Σ_{Rel}.

Note that in this solution we are crucially using the fact that we are composing the *same* Σ-protocol so that P can use any of the a_i committed in the 1st round to compute an honest transcript. This technique thus falls short as soon as we want to compose arbitrary Σ-protocols together. Nevertheless, this transformation turns to be useful for the case of different Σ-protocols.

(k, n)-equivocal commitment scheme. A (k, n)-equivocal commitment scheme allows a sender to compute n commitments $\mathtt{com}_1, \ldots, \mathtt{com}_n$ such that k of them are binding and $n - k$ are equivocal. We will use the language DH of DH tuples and we will implement a (k, n)-equivocal commitment scheme very efficiently under the DDH assumption as follows. In the commitment phase, the sender computes n tuples $T_1 = (g_1, A_1, B_1, X_1), \ldots, T_n = (g_n, A_n, B_n, X_n)$ and proves that k out of n tuples are *not* in DH. We show that this can be done using the classical [CDS94] (k, n)-proof of partial knowledge that can be obtained starting with a Σ-protocol Σ^{ddh} for DH.

We then use the well known [DG03, CV05, CV07, HL10] fact that Σ-protocols can be used to construct an Instance-Dependent Binding Commitment scheme, where the sender can equivocate if he knows the witness for the instance. Thus, each tuple T_i can be used to compute an instance-dependent binding commitment $\mathtt{com}_i$ using Σ^{ddh}. $\mathtt{com}_i$ will be equivocal if T_i was indeed a DH tuple, it will be binding otherwise. Because the sender proves that k tuples are not in DH, it holds that there are at least k binding commitment. Hiding follows from the WI property of [CDS94] and the HVZK of Σ^{ddh}. Commitment and decommitment can be completed in 3 rounds.

The case of different Σ-protocols. We now consider the case where we want to compose $\Sigma_1, \ldots, \Sigma_n$ for possibly different relations. Our (k, n)-equivocal commitment does not help here because each a_i is specific to protocol Σ_i, and cannot be arbitrarily mixed and matched once the k witnesses are known.

For this case we thus use a different trick. We ask the prover to commit to each a_i twice, once using a binding commitment and once using an equivocal commitment. This again can be very efficiently implemented from the DDH assumption as follows. For each i, P generates tuples T^0_i and T^1_i, that are such that at most one can be a DH tuple. It then commits to a_i twice using the instance-dependent binding commitment associated to tuple T^0_i and tuple T^1_i. Because at most one of the two tuples is a DH tuple, at most one of the commitments of a_i can be later equivocated. Thus the 1st round of our transformation consists of 2 commitments of a_i for $1 \leq i \leq n$.

In the 3rd round, when P receives instances $x_1, \ldots, x_n$ and k witnesses, he proceeds at follows. For each i, if P knows the witness for x_i, he will open the binding commitment for position i, and compute z_i using the honest prover procedure of Σ_i. Instead, if P does not have a witness for x_i, he will compute a new $\tilde{a}_i, z_i$ using the simulator on input x_i, c and open the equivocal commitment in position i. At the end, for each position i, one commitment has remained unopened.

This mechanism allows an honest prover to complete the proof with the knowledge of only k witnesses. However, what stops a malicious prover to always open the equivocal commitments and thus complete the proof without knowing any of the witnesses?

We avoid this problem by requiring P to prove that, among the n tuples corresponding to the unopened commitments, at least k out of n tuples are DH tuples. This directly means that k of the opened commitments were constructed over non-DH tuples, and therefore are binding.

Now note that proving this theorem requires an (k, n)-proof of partial knowledge in order to implement Σ^{ddh}, where the instance to prove, i.e., the tuple that will be unopened, is known only in the 3rd round when P knows for which instances he is able to open a binding commitment. Here we crucially use the (k, n)-proof of partial knowledge for the same Σ-protocol developed above making sure to first run our compiler that strengthen Σ^{ddh} with respect to statements adaptively selected by a malicious prover.

6.1.3 Comparison with the State of the Art

In Table 6.1 we compare our results with the relevant related work. We consider [LS90], a 3-round public-coin WIPoK that is *fully adaptive-input* and that works for any $\mathcal{NP}$ language. We also consider [CDS94] that proposed efficient 3-round public-coin WI proofs of partial knowledge (though, without supporting any adaptivity). Finally, we consider [CPS+16a] since it was the only work that faced the problem of combining together efficiency and some form of delayed-input instances.

The last row refers to our main result that allows to postpone knowledge of all the instances to the last round.

	Assumption	Adaptive WI	Adaptive PoK	$\mathcal{NP}$ Reduction
LaSh90 [LS90]	OWP	k out of n (all adaptive)	k out of n (all adaptive)	Yes
CDS94 [CDS94]	/	/	/	No
CPSSV16 [CPS+16a]	/	1 out of 2 (1 adaptive)	/	No
This Work (main result)	DDH	k out of n (all adaptive)	k out of n (all adaptive)	No

Table 6.1: Comparison with previous work.

The second column refers to the computational assumptions needed by [LS90] (i.e., one-way permutations) and our main result (i.e., DDH assumption). The third column specifies the type of WI depending on the adaptive selection of the instances from the adversarial verifier. The fourth column specifies the soundness depending on the adaptive selection of the instances from the adversarial prover.

6.1.4 Online/Offline Computation

The main feature of our constructions is the property of being delayed-input and thus the prover can compute the first round without knowing instances in an *offline phase*. When interacting with the verifier (the *online phase*), the prover sends the precomputed first round and computes only the third round of the protocol thus greatly improving the efficient of the protocol. As we have already pointed out, the LaSh construction enjoys a similar property and, instead, the construction of [CDS94] requires knowledge of the instances from the start of the protocol and so there is no offline phase.

In Table 6.2, we compare the computational effort of the prover in the online phase of our constructions with the computational effort of [CDS94, LS90] and the of the online phase of [CPS⁺16a] for proofs of partial knowledge of discrete logarithms. In the online phase of LaSh, the prover computes an $\mathcal{NP}$ reduction. For the remaining constructions, we report the number of modular[3] exponentiations that are computed by the prover in the online phase.

As far as the construction of [CDS94] is concerned, it is easy to see that $2n-k$ exponentiations are performed by the prover for the (n,k)-proof of partial knowledge.

	$(1,2)$ DLogs	(k,n) DLogs
LaSh	$\mathcal{NP}$-reduction	$\mathcal{NP}$-reduction
CDS94	3 exps	$2n-k$ exps
CPSSV16	4 exps	/
Our PoK and Adaptive-Input WI	2 exps	$2(n-k)$ exps
Our Adaptive-Input PoK and Adaptive-Input WI	4 exps	$4(n-k)$ exps

Table 6.2: Prover's computational effort in proofs of partial knowledge of discrete logarithms.

For the construction of [CPS⁺16a], we remind the reader that one instance, called the *original* instance, is known from the start of the protocol and the other, called the *adaptive* instance, is made available to the prover only after the first round has been completed along with the witness for one of the instances. In the $(1,2)$-proof of partial knowledge of [CPS⁺16a], the first round requires three exponentiations as the prover needs to run the simulator and the prover's algorithm of Schnorr's Σ-protocol. If the witness for the original instance is received, the prover needs to perform two extra exponentiations whereas if the witness received is for the adaptive instance then the prover can compute the third round by performing only one exponentiation. Of the five exponentiations required in the worst case, one can be precomputed (the one relative to the prover's algorithm of Schnorr's Σ-protocol) and thus the prover of [CPS⁺16a] needs to compute four exponentiations in the online phase. The last two rows of the table report on the number of exponentiations performed by the prover in two instantiations of the general construction of Section 6.4: the first is obtained by plugging in Schnorr's protocol (this only gives PoK) and the second is obtained by plugging in the adaptive-input 2-special sound protocol for discrete logarithm resulting by applying the compiler of Section 6.3.2 to Schnorr's protocol. For a detailed analysis on the number of exponentiations required by our two instantiations we refer the reader to Section 6.4.3.

6.2 Preliminaries

In this work we consider the notion of *threshold relation*. Let $\mathsf{Rel}_1, \ldots, \mathsf{Rel}_n$ be polynomial-time relations. A threshold relation with threshold k is a polynomial-time relation as well, with the following form

$$\begin{aligned}\mathsf{Rel}_k = \{&(x_1, \ldots, x_n), ((w_1, b_1) \ldots, (w_k, b_k)) : \\ &1 \le b_1 < \cdots < b_k \le n \text{ and } (x_{b_i}, w_i) \in \mathsf{Rel}_i, \text{ for } i = 1, \ldots, k\}.\end{aligned}$$

Roughly, a threshold relation with threshold k contains a set of n $\mathcal{NP}$ statements $x_1, \ldots, x_n$, and a set of witnesses $w_1, \ldots, w_k$ with $k \le n$, in which each w_i represents a valid witness for one and only one statement $x_{d_i} \in \hat{L}_{\mathsf{Rel}_{d_i}}$ with $d_i \in \{1, \ldots, n\}$.

[3]We will omit the word *modular* from now on.

6.2.1 Three rounds and public coins

The main results of this chapter will concern pairs $(\mathcal{P}, \mathcal{V})$ of interactive machines that interact for exactly three rounds with $\mathcal{P}$ sending the first message and with $\mathcal{V}$'s only message consisting solely of coin tosses. These pairs are called *three-round public-coin protocols* and have been object of intensive studies. Unless otherwise specified, whenever we say protocol, we mean three-round public-coin protocol. This class includes Σ-protocols [CDS94], that are widely used in practice, have been designed for several useful languages and, moreover, they are easy to work with as already shown in many transforms [DG03, MP03, Vis06, CDV06, BPSV08, YZ07, OPV10, Lin15, CPSV16].

We usually denote the *transcript* of an execution of a protocol $(\mathcal{P}, \mathcal{V})$ by a triple of messages (a, c, z), where a and z are sent by $\mathcal{P}$ and c, the *challenge*, is $\mathcal{V}$'s only message. We say that a transcript is *accepting* if $\mathcal{V}$ outputs 1. Two accepting transcripts (a, c, z) and (a', c', z') with the same common input x constitute a *collision for* x if $a = a'$ and $c \neq c'$.

We next present a generalization of the classical notion of a Σ-protocol.

Definition 22. *A protocol* $(\mathcal{P}, \mathcal{V})$ *is a* t-Σ-protocol *for polynomial-time relation* Rel *if it enjoys the following properties:*

- Completeness. *For every* $(x, w) \in \mathsf{Rel}$*, it holds that*

$$\text{Prob}\left[\, \langle \mathcal{P}(w), \mathcal{V}\rangle(x) = 1 \,\right] = 1.$$

- t-Special Soundness. *There exists a PPT algorithm* $\mathsf{Extract}$ *that, on input* x *and* t *transcripts such that any two of them constitute a collision for* x*, outputs a witness* w *for* x*.*

- Special Honest Verifier Zero Knowledge (SHVZK). *There exists a PPT* simulator *algorithm* Sim *that, on input an instance* $x \in L$ *and a challenge* c*, outputs* (a, z) *such that* (a, c, z) *has the same distribution of transcripts obtained when* $\mathcal{V}$ *sends* c *as challenge and* $\mathcal{P}$ *runs on common input* x *and any private input* w *such that* $(x, w) \in \mathsf{Rel}$.

It is easy to observe that the notion of a 2-Σ-protocol coincides with the notion of a Σ-protocol as introduced by [CDS94]. Indeed, we will refer to 2-special soundness as special soundness, thus conforming with the terminology of [CDS94]. We also stress that SHVZK as defined above corresponds to the notion of *Perfect* SHVZK as distinct from *Computational* SHVZK. This latter notion has also been studied in the literature in the context of Σ-protocols [GMY06a] but it will not be considered in this chapter.

SHVZK is a weaker requirement than Zero Knowledge; nonetheless, it implies non-trivial security against adversarial verifiers.

Theorem 8 ([CDS94])**.** *Let* Π *be a protocol that enjoys completeness and SHVZK for relation* Rel*. Then* Π *is Perfect WI.*

In a t-Σ-protocol security for $\mathcal{P}$ is unconditional. The following result implies instead that the challenge length acts as a security parameter for $\mathcal{V}$.

Theorem 9. *Let* Π *be a protocol for polynomial-time relation* Rel *that is* t*-special sound for some polynomially bounded* t*. Then* Π *is a proof of knowledge with knowledge error negligible in the challenge length.*

Proof. The proof of this theorem strictly follows the proof of Theorem 1 of [Dam10] in which t-special sound protocols are considered instead of 2-special sound protocols. $\square$

The following theorem says that the challenge length can be increased by simple parallel repetition.

Theorem 10. *[CDS94, Dam10] Let Π be a t-Σ-protocol for polynomial-time relation* Rel *with challenge length l. The k-wise parallel composition of Π is a Σ-protocol for* Rel *with challenge length $k \cdot l$.*

6.2.2 Delayed-input protocols

In this section, we present the notion of a *delayed-input* three-round public-coin protocol and give security notions both for the prover and the verifier.

Definition 23 (Delayed-input [CPS+16a])**.** *A* delayed-input *protocol for polynomial-time relation* Rel *is a protocol $(\mathcal{P}, \mathcal{V})$ in which the first message of $\mathcal{P}$ can be computed on input only the length, ℓ, of the common input in unary notation.*

For simplicity in the rest of the chapter, we will drop input 1^ℓ when describing the prover of a delayed-input protocol.

Since in a delayed-input protocol the input is not fixed at the onset of the protocol, it could be adversarially chosen by the prover or by the verifier based on the messages exchanged. For example, the special soundness of a Σ-protocol guarantees extraction from collisions only if the transcripts are for the same input and this might not be the case if the statement to be proved is chosen adaptively by the prover depending on the challenge received. We thus introduce a stronger notion that we call *adaptive-input t-special soundness.* Roughly speaking, we require that it is possible to extract witnesses from a collision even if the t accepting transcripts are for different inputs.

Definition 24. *A delayed-input protocol Π for relation* Rel *enjoys* adaptive-input t-special soundness *if there exists an efficient algorithm that, on input accepting transcripts $((a, c_1, z_1), \ldots, (a, c_t, z_t))$ for inputs $x_1, \ldots, x_t$, respectively, with $c_i \neq c_j$ for $1 \leq i < j \leq t$, outputs witnesses $(w_1, \ldots, w_t)$ such that $(x_i, w_i) \in \mathsf{Rel}$ for $i = 1, \ldots, t$.*

Adaptive-input 2-special soundness is closely related to the notion of an *adaptive-input proof of knowledge.* A protocol $\Pi = (\mathcal{P}, \mathcal{V})$ is an adaptive-input proof of knowledge if the proof of knowledge property holds even if the adversarial prover $\mathcal{P}^\star$ can choose the statement adaptively. To formally define the concept, we introduce, for every adversarial prover $\mathcal{P}^\star$, the family $\mathsf{aPoK}^{\mathcal{P}^\star} = \{\mathsf{aPoK}_\lambda^{\mathcal{P}^\star}\}_\lambda$ of probability distributions over $\{0,1\}^\star \cup \{\perp\}$. Distribution $\mathsf{aPoK}_\lambda^{\mathcal{P}^\star}$ assigns to $x \in \{0,1\}^\star$ the probability that $\mathcal{P}^\star$ concludes an accepting interaction with challenge length λ with the honest verifier $\mathcal{V}$ for input x; the symbol $\perp$ is instead assigned the probability that $\mathcal{V}$ rejects an interaction with challenge length λ with $\mathcal{P}^\star$.

Definition 25. *A delayed-input protocol $\Pi = (\mathcal{P}, \mathcal{V})$ is an* adaptive-input proof of knowledge with knowledge error $\kappa(\cdot)$ *for* Rel *if there exists an oracle machine* $\mathsf{AExtract}$ *such that, for all adversarial provers $\mathcal{P}^\star$ with* $\mathrm{Prob}\left[\, \langle \mathcal{P}^\star, \mathcal{V}\rangle(1^\lambda) = 1 \,\right] = p(\lambda) > \kappa(\lambda)$, $\mathsf{AExtract}^{\mathcal{P}^\star}(1^\lambda)$ *outputs pairs (x, w) with $x \in \{0,1\}^\star \cup \{\perp\}$ and*

- *whenever $x \neq \perp$, it holds that $(x, w) \in \mathsf{Rel}$;*
- *the family of probability distributions of the first output of $\mathsf{AExtract}^{\mathcal{P}^\star}$ is statistically close to $\mathsf{aPoK}^{\mathcal{P}^\star}$;*
- *there exists constant c such that $\mathsf{AExtract}^{\mathcal{P}^\star}$ stops within expected number of steps $\frac{\lambda^c}{p(\lambda) - \kappa(\lambda)}$.*

We have the following theorem.

Theorem 11. *Let Π be a delayed-input protocol. If Π is an adaptive-input t-special sound protocol for relation Rel, for some polynomially bounded t, then Π is an adaptive-input proof of knowledge with knowledge error $\kappa(\lambda) = (t-1)2^{-\lambda}$.*

Proof. We give the proof for $t = 2$. Extension to $t > 2$ is straightforward.

In order to prove that Π is an adaptive-input PoK for Rel, we consider the following $\mathsf{AExtract}^{\mathcal{P}^\star}$. $\mathsf{AExtract}^{\mathcal{P}^\star}$, upon receiving a from $\mathcal{P}^\star$, picks and sends a random challenge $c \leftarrow \{0,1\}^\lambda$ to $\mathcal{P}^\star$. Upon receiving z and the theorem x, $\mathsf{AExtract}$ checks if (a, c, z) is accepting for x. If $\mathcal{V}$ rejects, $\mathsf{AExtract}$ outputs $\perp$ and stops, otherwise it continues as follows. The extractor $\mathsf{AExtract}^{\mathcal{P}^\star}$ now rewinds $\mathcal{P}^\star$ by sending a new challenge $c' \leftarrow \{0,1\}^\lambda$, with $c \neq c'$, until another accepting transcript (a, c', z'), with respect to a potentially different input x', is provided by $\mathcal{P}^\star$. We now observe that the two accepting transcripts share the same first round and have different challenges, therefore by the adaptive-input 2-special soundness of Π the witness for x and x' can be computed in time $\mathsf{poly}(\lambda)$. At this point $\mathsf{AExtract}^{\mathcal{P}^\star}$ just outputs x and w such that $(x, w) \in \mathsf{Rel}$.

We now analyze the running time of $\mathsf{AExtract}$. Let $p^\star(\lambda) > 2^{-\lambda}$ be the probability that $\mathcal{P}^\star$ provides an accepting transcript for a challenge of length λ. Then the probability that, for a randomly selected challenge $c' \neq c$, $\mathcal{P}^\star$ provides an accepting transcript conditioned on the fact that c is accepting is $p^\star(\lambda) - 2^{-\lambda}$. Therefore the expected number of rewinds is $\frac{1}{p^\star(\lambda)-2^{-\lambda}}$. $\square$

The notion of an *adaptive-input WI* formalizes security of the prover with respect to an adversarial verifier $\mathcal{A}$ that adaptively chooses the input instance to the protocol; that is, after seeing the first message of the prover. We consider the computational version of this notion and therefore protocols will have a security parameter λ for the prover. More specifically, for a delayed-input protocol Π, we consider game $\mathsf{ExpAWI}_{\Pi,\mathcal{A}}$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ in which the instance x and two witnesses w_0 and w_1 for x are chosen by $\mathcal{A}$ *after* seeing the first message of the protocol played by the challenger. The challenger then continues the game by randomly selecting one of the two witnesses, w_b, and by computing the third message by running the prover's algorithm on input the instance x, the selected witness w_b and the challenge received from the adversary. The adversary wins the game if she can guess which of the two witnesses was used by the challenger.

We now define the adaptive-input WI experiment $\mathsf{ExpAWI}_{\Pi,\mathcal{A}}(\lambda, \mathsf{aux})$. This experiment is parameterized by a delayed-input protocol $\Pi = (\mathcal{P}, \mathcal{V})$ for a relation Rel and by PPT adversary $\mathcal{A}$. The experiment has as input the security parameter λ and auxiliary information aux for $\mathcal{A}$.

$\mathsf{ExpAWI}_{\Pi,\mathcal{A}}(\lambda, \mathsf{aux})$:

1. $\mathcal{C}$ randomly selects coin tosses r and runs $\mathcal{P}$ on input $(1^\lambda; r)$ to obtain a;
2. $\mathcal{A}$, on input a and aux, outputs instance x, witnesses w_0 and w_1 such that $(x, w_0), (x, w_1) \in \mathsf{Rel}$, challenge c and internal state $\mathtt{state}$;
3. $\mathcal{C}$ randomly selects $b \leftarrow \{0,1\}$ and runs $\mathcal{P}$ on input (x, w_b, c) and the randomness used at the first round thus obtaining z;
4. $b' \leftarrow \mathcal{A}((a, c, z), \mathsf{aux}, \mathtt{state})$;
5. if $b = b'$ then output 1 else output 0.

We set $\mathsf{AdvAWI}_{\Pi,\mathcal{A}}(\lambda, \mathsf{aux}) = \left|\text{Prob}\left[\ \mathsf{ExpAWI}_{\Pi,\mathcal{A}}(\lambda, \mathsf{aux}) = 1\ \right] - \frac{1}{2}\right|$.

Definition 26 (Adaptive-Input Witness Indistinguishability)**.** *A delayed-input protocol* Π *is* adaptive-input WI *if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that for any* $\mathsf{aux} \in \{0,1\}^*$ *it holds that* $\mathsf{AdvAWI}_{\Pi,\mathcal{A}}(\lambda, \mathsf{aux}) \leq \nu(\lambda)$.

6.2.3 The DDH assumption

Let $\mathcal{G}$ be a cyclic group, g generator of $\mathcal{G}$ and let A, B and X be elements of $\mathcal{G}$. We say that (g, A, B, X) is a *Diffie-Hellman tuple* (a *DH tuple*, in short) if $A = g^\alpha, B = g^\beta$ for some integers $0 \leq \alpha, \beta \leq |\mathcal{G}| - 1$ and $X = g^{\alpha\beta}$. If this is not the case, the tuple is called *non-DH.* To verify that a tuple is DH, it is sufficient to have the discrete log α of A to the base g and then to check that $X = B^\alpha$. We thus define the polynomial-time relation $\mathsf{DH} = \{((g, A, B, X), \alpha) : A = g^\alpha \text{ and } X = B^\alpha\}$ of the DH tuples.

The *Decisional Diffie-Hellman* assumption (the DDH assumption) posits the hardness of distinguishing a randomly selected DH tuple from a randomly selected non-DH tuple with respect to a *group generator* algorithm. For sake of concreteness, we consider the specific group generator GG that, on input 1^λ, randomly selects a λ-bit prime p such that $q = (p-1)/2$ is also prime and outputs the (description of the) order q group $\mathcal{G}$ of the quadratic residues modulo p along with a random generator g of $\mathcal{G}$.

Assumption 1 (DDH Assumption)**.** *For every probabilistic polynomial-time algorithm* $\mathcal{A}$ *there exists a negligible function* ν *s.t.*

$$\Big| \mathrm{Prob}\Big[(\mathcal{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda); \alpha, \beta, \gamma \leftarrow \mathbb{Z}_q : \mathcal{A}((\mathcal{G}, q, g), g^\alpha, g^\beta, g^\gamma) = 1 \Big] - \\ \mathrm{Prob}\Big[(\mathcal{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda); \alpha, \beta, \gamma \leftarrow \mathbb{Z}_q : \mathcal{A}((\mathcal{G}, q, g), g^\alpha, g^\beta, g^{\alpha\beta}) = 1 \Big] \Big| \leq \nu(\lambda).$$

1-non-DDH. We define polynomial time relation $\mathsf{1nDH}$ as $\mathsf{1nDH} = \{((g, A, B, X), \alpha) : A = g^\alpha \text{ and } X = g \cdot B^\alpha\}$. Thus a *1-non-DH* tuple is a tuple $T = (g, A, B, X)$ such that $A = g^\alpha$, $B = g^\beta$ and

$$X = g \cdot B^\alpha = g^{\alpha \cdot \beta + 1}$$

Under the DDH assumption random 1-non-DH tuples are indistinguishable from random non-DH tuple. Indeed, let $T = (g, A, B, X)$ be any tuple and consider tuple $T' = (g, A, B, g \cdot X)$. Then we have that if T is a randomly selected DH tuple, then T' is a randomly selected 1-non-DH tuple; whereas, if T is a randomly selected non-DH tuple then T' is statistically close to a randomly selected non-DH tuple. By transitivity, we have that, under the DDH assumption, randomly selected 1-non-DH tuples are indistinguishable from randomly selected DH tuples We can thus state the following lemma (first proved as Lemma 3.1 in [HKR$^+$14]).

Lemma 7. *[HKR$^+$14] Under the DDH assumption, for every probabilistic polynomial-time algorithm* $\mathcal{A}$ *there exists a negligible function* ν *s.t.*

$$\Big| \mathrm{Prob}\Big[(\mathcal{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda); \alpha, \beta \leftarrow \mathbb{Z}_q : \mathcal{A}((\mathcal{G}, q, g), g^\alpha, g^\beta, g^{\alpha\beta+1}) = 1 \Big] - \\ \mathrm{Prob}\Big[(\mathcal{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda); \alpha, \beta \leftarrow \mathbb{Z}_q : \mathcal{A}((\mathcal{G}, q, g), g^\alpha, g^\beta, g^{\alpha\beta}) = 1 \Big] \Big| \leq \nu(\lambda).$$

A Σ-protocol Π_k^{1nDH} for the threshold relation of $\mathsf{1nDH}$ can be constructed based on the Σ-protocol Π^{ddh} of [CDS94] to prove that k out of n tuples are DDH tuples as follows. Specifically, let $\mathsf{1nDH}_k$ be the threshold relation for $\mathsf{1nDH}$ defined as follows:

$$\mathsf{1nDH}_k = \Big\{ \Big(((g_1, A_1, B_1, X_1), \ldots, (g_n, A_n, B_n, X_n)), ((\alpha_1, b_1) \ldots, (\alpha_k, b_k)) \Big) : \\ 1 \leq b_1 < \cdots < b_k \leq n \text{ and } A_{b_i} = g_{b_i}^{\alpha_i} \text{ and } X_{b_i} = g_{b_i} \cdot B_{b_i}^{\alpha_i}, \text{ for } i = 1, \ldots, k \Big\}.$$

We construct Σ-protocol $\Pi_k^{\mathsf{1nDH}} = (\mathcal{P}_k^{\mathsf{1nDH}}, \mathcal{V}_k^{\mathsf{1nDH}})$ as follows. On input tuples $(g_1, A_1, B_1, X_1), \ldots, (g_n, A_n, B_n, X_n)$, $\mathcal{P}_k^{\mathsf{1nDH}}$ and $\mathcal{V}_k^{\mathsf{1nDH}}$ construct tuples $(g_1, A_1, B_1, Y_1), \ldots, (g_n, A_n, B_n, Y_n)$ by setting $Y_i = X_i/g_i$, for $i = 1, \ldots, n$. Then, $\mathcal{P}_k^{\mathsf{1nDH}}$ and $\mathcal{V}_k^{\mathsf{1nDH}}$ simply run Σ-protocol Π^{ddh} on input the theorem $(g_1, A_1, B_1, Y_1), \ldots, (g_n, A_n, B_n, Y_n)$.

Theorem 12. *Protocol Π_k^{1nDH} is a Σ-protocol for the threshold relation,$\mathsf{1nDH}_k$, of $\mathsf{1nDH}$.*

Proof. The completeness follows from the completeness of Π^{ddh} and from the fact that $\mathcal{G}$ is a cyclic group with generator g. The special-soundness of Π^{1nDH} follows from the special soundness of Π^{ddh}. Indeed, a collision for Π^{1nDH} represents a collision for Π^{ddh} as well. Thus, the witnesses for k of the constructed tuples can be extracted. This part of the proof ends with the observation that k witnesses for the constructed tuples correspond to k witnesses for the input tuples.

If at least k of the input tuples are 1-non-DH then at least k of the *constructed* tuples (g_i, A_i, B_i, Y_i) are DH and the prover has a witness for it. On the other hand, if fewer than k of the input tuples are 1-non-DH then the constructed tuples contain fewer than k DH tuples.

The SHVZK property follows directly from the same property of Π^{ddh}. □

6.2.4 Instance-Dependent Binding Commitment

We define the notion of an Instance-Dependent Binding Commitment scheme associated with a polynomial-time relation Rel and describe a construction for all Rel admitting a Σ-protocol.

Definition 27 (Instance-Dependent Binding Commitment scheme)**.** *An* Instance-Dependent Binding Commitment *scheme (an* IDBC*, in short) for polynomial-time relation* Rel *with message space M is a quadruple of PPT algorithms* $(\mathsf{Com}, \mathsf{Dec}, \mathsf{Fake}_1, \mathsf{Fake}_2)$ *where:*

- Com *is the randomized* commitment *algorithm that takes as input an instance $x \in \hat{L}_{\mathsf{Rel}}$ and a message $m \in M$ and outputs* commitment $\mathtt{com}$ *and* decommitment $\mathtt{dec}$*;*
- Dec *is the* verification *algorithm that takes as input $x \in \hat{L}_{\mathsf{Rel}}$,* $\mathtt{com}, \mathtt{dec}$ *and $m \in M$ and decides whether m is the decommitment of* $\mathtt{com}$*;*
- Fake_1 *takes as input* $(x, w) \in \mathsf{Rel}$ *and outputs* commitment $\mathtt{com}$*, and* equivocation information $\mathtt{rand}$*;*
- Fake_2 *takes as input* $(x, w) \in \mathsf{Rel}$*, message $m \in M$, and pair* $(\mathtt{com}, \mathtt{rand})$ *and outputs* $\mathtt{dec}$*;*

that enjoy the following properties:

- Correctness*: for all $x \in \hat{L}_{\mathsf{Rel}}$, all $m \in M$, it holds that*

$$\text{Prob}\left[\, (\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{Com}(x, m) : \mathsf{Dec}(x, \mathtt{com}, \mathtt{dec}, m) = 1 \,\right] = 1.$$

- Binding*: for all $x \notin L_{\mathsf{Rel}}$ and for every commitment* $\mathtt{com}$ *there exists at most one message $m \in M$ for which there exists a valid decommitment* $\mathtt{dec}$*; that is, such that* $\mathsf{Dec}(x, \mathtt{com}, \mathtt{dec}, m) = 1$.
- Hiding*: for every receiver $\mathcal{A}$, for every auxiliary information* aux*, for all $x \in L_{\mathsf{Rel}}$ and all for $m_0, m_1 \in M$, it holds that*

$$\text{Prob}\left[\, b \leftarrow \{0,1\}; (\mathtt{com}, \mathtt{dec}) \leftarrow \mathsf{Com}(1^\lambda, x, m_b) : b = \mathcal{A}(\mathsf{aux}, x, \mathtt{com}, m_0, m_1) \,\right] \le \frac{1}{2}.$$

- Trapdoorness*: for all* $(x, w) \in \mathsf{Rel}$ *and* $m \in M$ *the following two distributions coincide*

$$\{(\texttt{com}, \texttt{rand}) \leftarrow \mathsf{Fake}_1(x, w); \texttt{dec} \leftarrow \mathsf{Fake}_2(x, w, m, \texttt{com}, \texttt{rand}) : (\texttt{com}, \texttt{dec})\}$$

$$\{(\texttt{com}, \texttt{dec}) \leftarrow \mathsf{Com}(x, m) : (\texttt{com}, \texttt{dec})\}.$$

IDBC from Σ-protocol. Our construction follows similar constructions of [Dam10, HL10, DN02]. Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a Σ-protocol for the polynomial-time relation Rel challenge length λ, and simulator S. We define IDBC $\mathsf{CS}^{\Pi} = (\mathsf{Com}^{\Pi}, \mathsf{Dec}^{\Pi}, \mathsf{Fake}_1^{\Pi}, \mathsf{Fake}_2^{\Pi})$ with message space $\{0, 1\}^{\lambda}$ as follows.

- Com^{Π} takes as input x and $m \in \{0, 1\}^{\lambda}$ and computes $(\texttt{com}, \texttt{dec})$ by running S on input (x, m).
- Dec^{Π} takes as input $x, \texttt{com}, \texttt{dec}, m$, runs $\mathcal{V}$ on instance x and transcript $(\texttt{com}, m, \texttt{dec})$ and outputs $\mathcal{V}$'s output.
- Fake_1^{Π} takes as input $(x, w) \in \mathsf{Rel}$, samples a random string ρ and runs $\mathcal{P}$ on input (x, w) and randomness ρ to get the 1st message a of Π. Fake_1^{Π} sets $\texttt{com} = \mathsf{a}$ and $\texttt{rand} = \rho$ and outputs $(\texttt{com}, \texttt{rand})$.
- Fake_2^{Π} takes as input $(x, w) \in \mathsf{Rel}$, message $m \in M$, and commitment $\texttt{com}$ and equivocation information $\texttt{rand}$ and runs $\mathcal{P}$ to get third-round message z of Π corresponding to first message $\mathsf{a} = \texttt{com}$ produced with randomness $\rho = \texttt{rand}$ and challenge $\mathsf{c} = m$. Fake_1^{Π} sets $\texttt{dec} = \mathsf{z}$ and outputs $\texttt{dec}$.

Theorem 13. CS^{Π} *is an IDBC for the polynomial-time relation* Rel.

Proof. The security proof relies only on the properties of Π. Correctness follows from the completeness of Π. Binding follows from the special soundness of Π. Hiding and Trapdoorness follow from the SHVZK and the completeness of Π. □

6.3 Adaptive-Input Special-Soundness of Σ-protocols

In this section we start by describing classical Σ-protocols that are not secure when the adversarial prover can choose the statement adaptively after seeing the verifier's challenge. We then give an efficient compiler that, on input a Σ-protocol belonging to the general class considered in [Mau15, CD98] and that includes our examples, outputs a Σ-protocol that is adaptive-input sound.

6.3.1 Adaptive-Input Insecure Delayed-input Σ-protocols

We start with the following well-known Σ-protocol $\Pi_{\mathsf{DH}} = (\mathcal{P}, \mathcal{V})$ for relation DH. On common input $T = (g, A, B, X)$ and private input α such that $A = g^{\alpha}$ and $X = B^{\alpha}$, the prover $\mathcal{P}$ sends $a = g^r$ and $x = B^r$ for randomly chosen $r \in Z_q$ (q denotes the size of the group $\mathcal{G}$). Upon receiving challenge c, $\mathcal{P}$ replies by sending $z = r + \alpha \cdot c$ to $\mathcal{V}$. $\mathcal{V}$ accepts if and only if $g^z = a \cdot A^c$ and $B^z = x \cdot X^c$.

We construct two accepting transcripts for Π_{DH}, one for each of two non-DH tuples. The two transcripts have the same first-round message and two different challenges (thus they constitute a collision) but, obviously, no witness can be extracted. Specifically, consider tuples $T_1 = (g, A = g^{\alpha}, B, X_1 = B^{\gamma_1})$ and $T_2 = (g, A = g^{\alpha}, B, X_2 = B^{\gamma_2})$ with $\gamma_1, \gamma_2 \neq \alpha$ (this guarantees that neither tuple is DH). We constructs the two transcripts by setting the common first-round message to $(a = g^r, x = B^s)$, for arbitrary $r \neq s$. Then we compute challenges c_1 and c_2 by setting $c_i = \frac{r-s}{\gamma_i - \alpha}$, for $i = 1, 2$; clearly, the two transcripts have different challenges. We complete the transcripts by computing the third-round messages z_1 and z_2 as $z_i = r + \alpha \cdot c_i$, for $i = 1, 2$. It is then easily see that both transcripts are accepting.

We remark that essentially the same reasoning can be used to prove that also adaptive-input soundness can be violated by a prover that is allowed to pick T after seeing c. Specifically, consider a malicious prover that first sends $(a = g^r, x = B^s)$, with $r \neq s$, and then it completes the protocol for adaptively chosen tuple $T = (g, A = g^\alpha, B, X = B^\gamma)$ with $\gamma = \alpha + \frac{r-s}{c}$ by sending $z = r + \alpha \cdot c$. Tuple T is non-DH (as $\gamma \neq \alpha$ by the choice of r and s) and the verifier is easily seen to accept. Similar arguments apply for the Σ-protocol of [MP03] for relation $\mathsf{Com} = \{((g, h, G, H, m), r) : G = g^r \text{ and } H = h^{r+m}\}$.

We next consider Schnorr's Σ-protocol [Sch89] for relation $\mathsf{DLog} = \{((\mathcal{G}, g, Y), y) : g^y = Y\}$. In Schnorr's protocol, the prover on input $(Y, y) \in \mathsf{DLog}$ starts by sending $a = g^r$, for a randomly chosen $r \in Z_q$ (q denotes the size of the group $\mathcal{G}$). Upon receiving challenge c, $\mathcal{P}$ replies by computing $z = r + yc$. $\mathcal{V}$ accepts (a, c, z) if and only if $g^z = a \cdot Y^c$. We show an efficient procedure that on input $Y_1 \in \mathcal{G}$ and challenges $c_1 \neq c_2$, outputs $Y_2 \in \mathcal{G}$ and accepting transcripts (a, c_1, z_1) and (a, c_2, z_2) for Y_1 and Y_2, respectively. This implies that, unless computing discrete log is easy, it is not possible to extract the discrete log of Y_1 from the two transcripts.

The first transcript (a, c_1, z_1) is constructed by running the SHVZK simulator of Schnorr's protocol. Then we set $Y_2 = Y_1^\beta \cdot g^\gamma$, where $\beta = c_1/c_2$ and γ is arbitrary in Z_q, and complete the second transcript by setting $z_2 = z_1 + c_2 \cdot \gamma$. We have that

$$\begin{aligned} g^{z_2} &= g^{z_1} \cdot g^{c_2 \cdot \gamma} \\ &= a \cdot Y_1^{c_1} \cdot g^{c_2 \cdot \gamma} \\ &= a \cdot Y_1^{c_2 \cdot \beta} \cdot g^{c_2 \cdot \gamma} \\ &= a \cdot [Y_1^\beta \cdot g^\gamma]^{c_2} \\ &= a \cdot Y_2^{c_2} \end{aligned}$$

and thus the second transcript is also accepting for Y_2.

6.3.2 A Compiler for Adaptive-Input Special Soundness

In this section we construct an adaptive-input special sound Σ-protocol Π_f^{a} for proving knowledge of the pre-image of a homomorphic function f. Our construction is based on the Σ-protocol $\Pi_f = (\mathcal{P}_f, \mathcal{V}_f)$ given in [CD98, Mau15] that generalizes and subsumes several Σ-protocols, including the ones by Schnorr [Sch89] and Guillou-Quisquater [GQ88], and the Σ-protocol for DH tuples [DH76]. Our construction works for the same class of *canonical* homomorphic functions f used in [CD98, Mau15].

More precisely, let $(\mathcal{G}, \star)$ and $(\mathcal{H}, \otimes)$ be two groups with efficient operations and let $f : \mathcal{G} \to \mathcal{H}$ be a one-way homomorphism from $\mathcal{G}$ to $\mathcal{H}$; that is, for all $x, y \in \mathcal{G}$, we have that $f(x \star y) = f(x) \otimes f(y)$ and, on input $x = f(w)$ for a randomly chosen w, it is infeasible to compute w' such that $f(w') = x$. We next describe Σ-protocol $\Pi_f = (\mathcal{P}_f, \mathcal{V}_f)$ for relation $\mathsf{Rel}_f = \{(x, w) : x = f(w)\}$. Here prover $\mathcal{P}_f$ and verifier $\mathcal{V}_f$ receive as input the descriptions of groups $\mathcal{G}$ and $\mathcal{H}$ along with $x \in \mathcal{H}$. In addition, $\mathcal{P}_f$ receives w such that $x = f(w)$ as a private input. The first-round message a is computed by $\mathcal{P}_f$ by randomly picking $r \in \mathcal{G}$ and by setting $a = f(r)$. The verifier's challenge is randomly selected from a subset C_f of $\{0, \ldots, |\mathcal{G}| - 1\}$. The last message z is computed by $\mathcal{P}_f$ by setting $z = r \star w^c$. Finally, $\mathcal{V}_f$ accepts if and only if $f(z) = a \otimes x^c$.

Completeness is easily seen to hold and it is also easy to see that Schnorr's [Sch89] and Guillou-Quisquater [GQ88] Σ-protocols as well as the one for DH tuples are special cases of Π_f. Therefore, since Schnorr's protocol is a special case, protocol Π_f does not enjoy adaptive-input 2-special soundness. For special soundness, Theorem 3 of [Mau15] gives necessary conditions for witness w such that $x = f(w)$ to be extracted from a a collision (a, c_1, z_1) and (a, c_2, z_2) for common input x. Specifically, this is possible provided that integer y and $u \in \mathcal{G}$ such that $\gcd(c_1 - c_2, y) = 1$ and $f(u) = x^y$ can be efficiently computed. We call a homomorphic function with the above property *canonical*.

From Π_f to Π_f^{a}. By looking back at the proof that the Σ-protocol for DH does not enjoy adaptive-input 2-special soundness, we notice that it was crucial for the malicious prover to be able to pick the first message (a, x) so that (g, a, B, x) was a non-DH tuple. Indeed, if the prover was forced to pick a DH-tuple then it could not have made the honest verifier accept. We next show that this holds in general: if we force the prover to correctly compute the first message then Π_f is adaptive-input 2-special soundness; for Π_f this means to prove knowledge of r such that $a = f(r)$. This is easily achieved by having a second parallel instance of Π_f run on input a. We observe that the second instance of Π_f is not subject to an adaptive-input attack since in a collision the two transcripts share the first message a of the first instance of Π_f thus fixing the input of the second instance. Thus, in sums, protocol Π_f^{a} on common input x consists of the parallel execution of two instances of Π_f: the first with common input x and the second with common input the first-round message a of the first instance. The verifier of Π_f^{a} sends the same challenge to both instances and accepts if and only if, in both instances, the verifier of Π_f accepts. For reasons that will be clear in the proof, we set the challenge space of Π_f^{a} to $\mathsf{C}_f^{\mathsf{a}} = \mathsf{C}_f \setminus \{0\}$. The following observations are straightforward.

1. If the challenge space C_f consists of only one challenges then Π_f is vacuously adaptive-input 2-special sound and thus we consider $\Pi_f^{\mathsf{a}} = \Pi_f$.

2. If the challenge space C_f consists of two challenges and one of them is 0 then we have $\Pi_f^{\mathsf{a}} = \Pi_f$ and $\mathsf{C}_f^{\mathsf{a}} = \mathsf{C}_f - \{0\}$. We observe that, by definition, Π_f^{a} enjoys the property of adaptive-input 2-special soundness.

3. In general, if C_f contains 0, then the output protocol of the compiler has challenge space smaller than C_f, and therefore has also worst knowledge error than the starting protocol. This can be easily avoided by using Theorem 10 in order to amplify the challenge space of Π_f before using our compiler. We recall that it is important to preserve the knowledge error to preserve the PoK property of the input protocol[4].

Let us now prove that Π_f^{a} enjoys adaptive-input 2-special soundness whenever Π_f enjoys special soundness (and we are not in Case 1 or 2 described before). Suppose we have two accepting transcripts (a, c_1, z_1) and (a, c_2, z_2) of Π_f^{a} for inputs x_1 and x_2 with $c_1 \neq c_2$. By special soundness of the second instance of Π_f, we can extract the randomness r used to compute the first message a of the first instance of Π_f from the two sub-transcripts of the second instance of Π_f. We conclude the proof by showing that it is possible to compute w such that for $x = f(w)$ from one accepting transcript (a, c, z) of Π_f for x, provided that r such that $f(r) = a$ is available. We also assume that, y and u such that $f(u) = x^y$ and $\gcd(y, c) = 1$ are available (we use the same hypothesis of Theorem 3 of [Mau15]). Note that this is the main reason why we require $c \neq 0$. By using the extended GCD algorithm, we can compute α and β such that $y \cdot \alpha + c \cdot \beta = 1$.

[4]We observe that the Theorem 9 cannot be directly applied to the Σ-protocol described in this section because of the different notation used to describe the challenge space of a Σ-protocol.

Finally, we set $w = u^{\alpha} \star (r^{-1} \star z)^{\beta}$. Now observe that $f(z) = a \otimes x^{c}$ and this implies that $f(r^{-1} \star z) = x^{c}$. Therefore, $f(w) = f(u^{\alpha} \star (r^{-1} \star z)^{\beta}) = f(u)^{\alpha} \otimes f(z \star r^{-1})^{\beta} = x^{y\alpha} \otimes x^{\beta c} = x$ that proves that $f(w) = x$.

We have thus proved the following theorem.

Theorem 14. *Let f be a canonical one-way homomorphism. Then there exists a Σ-protocol Π_f^{a} for Rel_f that enjoys adaptive-input 2-special soundness.*

6.3.3 On the Adaptive-Input Soundness of [CPS+16a]'s Transform

Ciampi et al. in [CPS+16a] give a compiler that takes as input two Σ-protocols, Π_0 and Π_1 for languages L_0 and L_1, and constructs a new Σ-protocol Π^{OR} for $L_0 \vee L_1$ in which the instance for language L_1 is needed by the prover only in the 3rd round. The compiler requires Π_1 to be delayed input. In this section we show that if Π_1 enjoys adaptive-input 2-special soundness, then so does Π^{OR}. We start by giving a succinct description of the main building block used by [CPS+16a].

t-Instance-Dependent Binding Commitment. In a t-Instance-Dependent Binding Commitment (a t-IDBC) scheme for polynomial-time relation Rel, one party can commit a message m from a predefined message space M with respect to an instance x. The commitment produced is hiding and binding unless a witness w such that $(x, w) \in \mathsf{Rel}$; in this case, the commitment can be equivocated.

More precisely, a t-IDBC scheme consists of a triple of PPT algorithms $(\mathsf{TCom}, \mathsf{TDec}, \mathsf{TFake})$ with the following syntax. The *commitment* algorithm TCom takes as input an instance x and a message m and returns a commitment $\mathtt{com}$ of m with respect to x along with a decommitment $\mathtt{dec}$. The decommitment algorithm TDec takes as input $\mathtt{com}$, $\mathtt{dec}$ and m and x and verifies whether $\mathtt{dec}$ is an opening of $\mathtt{com}$ as m with respect to x. TFake is the *equivocation* procedure that, given a witness for an instance x, a commitment $\mathtt{com}$ of message m with respect to x along with the random coin tosses used to produce it and message m' outputs a decommitment $\mathtt{dec}'$ of $\mathtt{com}$ as m'. It is required that a t-IDBC is *correct* (honestly computed commitments are correctly decommitted), *hiding* (honestly computed commitments hide the message), *trapdoor* (a fake commitment is indistinguishable from an honestly computed commitment) and *t-Special Extractable*. The property of t-Special Extractability informally says that if the sender opens the same commitment in t different ways, then it is possible to efficiently extract the witness w. A construction of a 2-IBTC scheme that is perfect hiding, perfect trapdoor and 2-Special Extractable from a special type of Σ-protocols is given in [CPS+16a].

The Construction of [CPS+16a]

Let Rel_0 be a relation admitting a t-IDBC scheme with $t = 2$ and let Rel_1 be a relation admitting an delayed-input Σ-protocol Π_1; we denote by S_1 the associated simulator for the honest verifier zero knowledge. We next describe the Σ-protocol $\Pi^{\mathsf{OR}} = (\mathcal{P}^{\mathsf{OR}}, \mathcal{V}^{\mathsf{OR}})$ of [CPS+16a] for the OR relation:

$$\mathsf{Rel}^{\mathsf{OR}} = \Big\{((x_0, x_1), w) : ((x_0, w) \in \mathsf{Rel}_0 \wedge x_1 \in \hat{L}_{\mathsf{Rel}_1}) \text{ OR } ((x_1, w) \in \mathsf{Rel}_1 \wedge x_0 \in \hat{L}_{\mathsf{Rel}_0})\Big\}.$$

We assume that x_0 and the length of x_1 in unary are available from the onset of the protocol whereas x_1 and the witness w such that $((x_0, x_1), w) \in \mathsf{Rel}^{\mathsf{OR}}$ are given before the 3rd round. Obviously, the verifier does not get to learn w.

1. $\mathcal{P}^{\mathsf{OR}}$ executes the following steps:

1.1. pick random r_1 and compute the 1st round a_1 of the delayed-input Σ-protocol Π_1 using r_1 as random coin tosses;

1.2. compute a pair $(\mathtt{com}, \mathtt{dec}_1)$ of commitment and decommitment of a_1 with respect to x_0.

1.3. send $\mathtt{com}$ to $\mathcal{V}^{\mathsf{OR}}$.

2. $\mathcal{V}^{\mathsf{OR}}$ sends a random challenge c.

3. $\mathcal{P}^{\mathsf{OR}}$ on input $((x_0, x_1), c, (w, b))$ s.t. $(x_b, w) \in \mathsf{Rel}_b$ executes the following steps:

3.1. If $b = 1$,
   compute the 3rd round of Π_1, z_1, using as input (x_1, w, c);

3.2. send $(\mathtt{dec}_1, a_1, z_1)$ to $\mathcal{V}^{\mathsf{OR}}$;

3.3. If $b = 0$,
   run simulator S_1 on input x_1 and c obtaining (a_2, z_2);
   use trapdoor to compute decommitment $\mathtt{dec}_2$ of $\mathtt{com}$ as a_2;

3.4. send $(\mathtt{dec}_2, a_2, z_2)$ to $\mathcal{V}^{\mathsf{OR}}$.

4. $\mathcal{V}^{\mathsf{OR}}$ receives (a, z) and $\mathtt{dec}$ from $\mathcal{P}^{\mathsf{OR}}$ and accepts if and only if the following conditions are satisfied:

4.1. (a, c, z) is an accepting conversation for x_1;

4.2. $\mathtt{dec}$ is a valid decommitment of $\mathtt{com}$ for a message a.

Adaptive-Input Soundness of Π^{OR}

We now show that Π^{OR} preserves the adaptive-input special soundness of the underlying Σ-protocol.

Theorem 15. *If* Rel_0 *admits a* 2-IBTC *and* Rel_1 *admits a delayed-input adaptive-input special-sound* Σ*-protocol, then* Π^{OR} *is an adaptive-input special-sound* Σ*-protocol.*

Proof. The claim follows from the adaptive-input special soundness of the underlying Σ-protocol Π_1 and from the 2-Special Extractability property of the 2-IBTC scheme. More formally, consider an accepting transcript $(\mathtt{com}, c, (a, z, \mathtt{dec}))$ for input (x_0, x_1) and an accepting transcript $(\mathtt{com}, c', (a', z', \mathtt{dec}'))$ for input (x_0, x_1'), where $c' \neq c$ and x_1 is potentially different from x_1'. We observe that:

- if $a = a'$ then, by adaptive-input 2-special soundness of Π_1, there exists an efficient extractor $\mathsf{AExtract}$ that, given as input $((a, c, z), x_1)$ and $((a', c', z'), x_1')$, outputs w_1 and w_1' s.t. $(x_1, w_1) \in \mathsf{Rel}_1$ and $(x_1', w_1') \in \mathsf{Rel}_1$;

- if $a \neq a'$, then $\mathtt{dec}$ and $\mathtt{dec}'$ are two openings of $\mathtt{com}$ with respect to x_0 for messages $a \neq a'$; then we can obtain a witness w_0 for x_0 by the 2-Special Extractability of the 2-IBTC scheme.

□

A similar arguments can be used to show that if Rel_0 admits a 3-IBTC and Rel_1 admits a delayed-input Σ-protocol with adaptive-input special soundness, then Π^{OR} enjoys the adaptive-input proof of knowledge property.

6.4 Delayed-input three-round protocols for the threshold relation

In this section we show that if a relation Rel has a delayed-input Σ-protocol Π, then we can construct a delayed-input 3-round public-coin PoK Π_k for the k-threshold relation. Also, under the DDH assumption, Π_k is adaptive-input WI. If, in addition, Π is *adaptive-input 2-special sound*, then Π_k is an adaptive-input proof of knowledge. Let us now proceed more formally.

For a polynomial-time relation Rel, we define the *k-threshold* relation Rel_k as follows

$$\mathsf{Rel}_k = \Big\{ \Big((x_1, \dots, x_n), ((w_1, d_1), \dots, (w_k, d_k)) \Big) : 1 \le d_1 < \dots < d_k \le n \\ \text{and } (x_{d_i}, w_i) \in \mathsf{Rel}, \text{ for } i = 1, \dots, k \Big\}.$$

Let us give the intuition behind our construction of protocol Π_k for Rel_k. The prover of Π_k starts by randomly selecting k 1-non-DH tuples and $n-k$ DH tuples. Each tuple is used to commit to a random and independently selected first message of protocol Π by using the IBTC $\mathsf{CS} = (\mathsf{Com}, \mathsf{Dec}, (\mathsf{Fake}_1, \mathsf{Fake}_2))$ for the polynomial time relation $\mathsf{1nDH}$ (defined in Sec. 6.2.3). More specifically, algorithm Com is used to compute the commitments w.r.t. the DH tuples and algorithm Fake_1 to compute the commitments w.r.t. the 1-non-DH tuples. Note that Π is a delayed-input Σ-protocol and thus the inputs are not needed to complete this step. All commitments and tuples are sent to the verifier.

The verifier's challenge c and n inputs along with the witnesses for k of them are then made available to the prover that executes the following for each input, distinguishing between inputs for which a witness is available and inputs for which no witness has been provided. For each of the former, the prover associates one the commitments computed with respect to a 1-non-DH tuple and opens it by using Dec thus revealing a first-round message. The corresponding third-round message is computed by the prover by using the witness. Instead, for each input for which no witness is available, the prover runs the SHVZK simulator of Π on input the challenge c and obtains an accepting transcript (a, c, z). The prover then associates to the input a commitment computed with respect to a DH tuple and computes, by using the algorithm Fake_2, an opening of it as a. The opening and z are then sent to the verifier.

In sums, for each input, the verifier receives an accepting transcript whose first message is shown to have been committed to in the first round. To ensure that the prover cannot cheat and choose n DH tuples (that will allow him to open all commitments produced at the first round at his will) the prover and the verifier engage in parallel into an execution of Σ-protocol Π_k^{1nDH} described in Section 6.2.

1st round. $\mathcal{P}_k \Rightarrow \mathcal{V}_k$:

1. Set $(\mathcal{G}, p, g) \leftarrow \mathsf{GG}(1^\lambda)$.
2. Randomly select tuples $T_1 = (g_1, A_1, B_1, X_1), \dots, T_n = (g_n, A_n, B_n, X_n)$ of elements of $\mathcal{G}$ under the constraint that exactly k are 1-non-DH and $n-k$ are DH, along with $\alpha_1, \dots, \alpha_n$ such that $A_i = g_i^{\alpha_i}$, for $i = 1, \dots, n$.
3. Let $b_1 < \ldots < b_k$ denote the indices of the k 1-non-DH tuples and $\widetilde{b}_1, \dots, \widetilde{b}_{n-k}$ denote the indices of the $n-k$ DH tuples.
4. Randomly select coin tosses ρ and run the prover of Π_k^{1nDH} on input $\mathcal{T} = (T_1, \dots, T_n)$, witness $((\alpha_{b_1}, b_1), \dots, (\alpha_{b_k}, b_k))$ and randomness ρ thus obtaining message a^{1nDH}. Send a^{1nDH} to $\mathcal{V}^k$.

5. For $i = 1, \dots, n$:

   Randomly select coin tosses r_i and compute first-round message a_i of Π by running $\mathcal{P}$ on input r_i.

   Compute pair $(\mathtt{com}_i, \mathtt{dec}_i) \leftarrow \mathsf{Com}(a_i, T_i)$.

   Send $(T_i, \mathtt{com}_i)$ to $\mathcal{V}_k$.

2nd round. $\mathcal{V}_k \Rightarrow \mathcal{P}_k$: randomly select a challenge c and send it to $\mathcal{P}_k$.

3rd round. $\mathcal{P}_k \Rightarrow \mathcal{V}_k$:

1. Receive inputs $(x_1, \dots, x_n)$ and witnesses $(w_{d_1}, \dots, w_{d_k})$ for inputs $x_{d_1}, \dots, x_{d_k}$ (we denote by $\widetilde{d}_1, \dots, \widetilde{d}_{n-k}$ the indices of the inputs for which no witness has been provided).
2. Compute the third round of Π^{1nDH} using c as challenge to get z^{1nDH} and send it to $\mathcal{V}^k$.
3. Randomly select permutation σ of $\{1, \dots, k\}$ and, for $i = 1, \dots, k$, associate 1-non-DH tuple T_{b_i} with input $x_{d_{\sigma(i)}}$.
4. For $i = 1, \dots, k$:

   Use j as a shorthand for $d_{\sigma(i)}$.

   Compute z_j by running $\mathcal{P}$ on input (x_j, w_j), a_{b_i}, randomness r_{b_i} and challenge c.

   Set $M_j = (j, b_i, \mathtt{dec}_{b_i}, a_{b_i}, z_j)$.
5. Randomly select permutation τ of $\{1, \dots, n-k\}$ and, for $i = 1, \dots, n-k$, associate DH tuple $T_{\widetilde{b}_i}$ with input $x_{\widetilde{d}_{\tau(i)}}$.
6. For $i = 1, \dots, n-k$:

   Use j as a shorthand for $\widetilde{d}_{\tau(i)}$.

   Run simulator S on input x_j and c obtaining (a_j, z_j).

   Use trapdoor $\alpha_{\widetilde{b}_i}$ to compute decommitment $\mathtt{dec}_{\widetilde{b}_i}$ of $\mathtt{com}_{\widetilde{b}_i}$ as a_j.

   Set $M_j = (j, \widetilde{b}_i, \mathtt{dec}_{\widetilde{b}_i}, a_j, z_j)$.
7. For $j = 1, \dots, n$: send M_j to $\mathcal{V}^k$.

$\mathcal{V}_k$ accepts if and only if all the following conditions are satisfied:

1. $(a^{\mathsf{1nDH}}, c, z^{\mathsf{1nDH}})$ is an accepting transcript for $\mathcal{V}^{\mathsf{1nDH}}$ with input T;
2. all t_j's are distinct;
3. for $j = 1, \dots, n$: $\mathtt{dec}_{t_j}$ is a valid decommitment of $\mathtt{com}_{t_j}$ with respect to T_{t_j};
4. for $j = 1, \dots, n$: (a_j, c, z_j) is an accepting transcript for $\mathcal{V}$ with input x_j;

We observe that, clearly, Π_k is a delayed-input 3-round public coin and that Π_k has the same challenge length as Π. In the next sections we prove its security properties.

6.4.1 Proof of Knowledge

In this section we start by proving that Π_k is a proof of knowledge with knowledge error negligible in the length of the challenge. Then we prove that enjoys Π is adaptive-input 2-special soundness then Π_k is an adaptive-input proof of knowledge.

Theorem 16. *Protocol Π_k is a proof of knowledge for Rel_k with knowledge error negligible in the length of the challenge.*

Proof. The completeness property is straightforward from the completeness of Π_k^{1nDH} and Π and from the correctness and trapdoor properties of the Instance-Dependent Binding Commitment scheme used.

Now we proceed to prove that Π_k is N-special sound with $N = k(n-k+1)+1$. Therefore, since $n = \mathsf{poly}(\lambda)$ and $k \leq n$, we can invoke Theorem 9 to conclude the security proof. Specifically, we show that there exists an efficient extractor that, for any sequence $(x_1, \ldots, x_n)$ of n inputs and for any set of N accepting transcripts of Π_k that share the same first message and have different challenges, outputs the witnesses of k of the n inputs. The extractor is based on the following observations.

From any two (out of the N) accepting transcripts of Π^k, we can extract a collision for Π_k^{1nDH}. By the special soundness of Π_k^{1nDH}, the collision gives us the witnesses that k of the tuples used in the N transcripts are 1-non-DH. Without loss of generality, we assume that we obtain witnesses for $T_1, \ldots, T_k$ and we call them the *good* tuples. This implies that commitments $\mathtt{com}_1, \ldots, \mathtt{com}_k$ that appear in the shared first round of the N transcripts are opened to the same strings $a_1, \ldots, a_k$ in all the N transcripts. The k good tuples can be associated in each of the N transcripts to different inputs; however, if the same good tuple is associated with the same input x_i in two different transcripts then we obtain a collision for Π with input x_i and thus, by the special soundness of Π, it is possible to extract a witness for x_i. We call such inputs *good* as well. Notice that we have N transcripts and k good tuples and thus we have $k \cdot N$ associations of good tuple to inputs.

We next prove that the N transcripts contain collisions of protocol Π with good tuples for at least k inputs and thus at least k witnesses can be extracted. For the sake of contradiction, suppose that there are only $m \leq k-1$ good inputs and let us count the good-tuple-to-input association (which we know to be $k \cdot N$) by counting the contributions of the good inputs and of the non-good inputs, separately. Clearly, each of the m good inputs is associated with at most one good tuple for each of the N transcripts thus contributing $m \cdot N$ associations. Each of the remaining $n - m$ inputs is associated with each of the k good tuples in at most *one* of the N transcripts thus contributing an extra $(n-m) \cdot k$ associations. We have thus counted a total of $m \cdot N + (n-m) \cdot k$ associations. Since the number of associations is an increasing function of m and since $m \leq k-1$, we have

$$\begin{aligned} m \cdot N + (n-m) \cdot k &= m \cdot (N-k) + n \cdot k \\ &\leq (k-1) \cdot (N-k) + n \cdot k \\ &\leq k \cdot N - N + k \cdot (n-k+1) \\ &= k \cdot N - 1 \end{aligned}$$

Contradiction. □

Next we prove that if that adaptive-input security of Π is preserved by our construction. More precisely,

Theorem 17. *If Π is adaptive-input 2-special sound for Rel then Π_k is an adaptive-input proof of knowledge for Rel_k with knowledge error negligible in the length of the challenge.*

Proof. The delayed-input completeness of Π_k follows from the delayed-input completeness of protocol Π, the correctness and trapdoor property of the Instance-Dependent Binding Commitment scheme used and from the completeness of Π_k^{1nDH}. In order to conclude the security proof we just need to show that Π_k is adaptive-input 2-special sound which, by Theorem 11, is sufficient to prove that Π_k is an adaptive-input PoK with knowledge error $\kappa(\lambda) = 2^{-\lambda}$. The proof ends with the observation that $2^-\lambda$ represents a negligible function in the challenge length.

Lemma 8. *If* Π *is adaptive-input 2-special sound for* Rel*, then* Π_k *is adaptive-input 2-special sound for* Rel_k*.*

Proof. Let $T_1 = (a, c_1, z_1)$ and $T_2 = (a, c_2, z_2)$ with $c_1 \neq c_2$ be two accepting transcripts for input $(x_1^1, \dots, x_n^1)$ and $(x_1^2, \dots, x_n^2)$, respectively. First of all, observe that, by the special soundness of protocol Π_k^{1nDH}, it is possible to extract the witness certifying that k of the tuple $T_1, \dots, T_n$ appearing in the first message are 1-non-DH. Let us denote by $b_1, \dots, b_k$ the indices of the 1-non-DH tuples. This implies that commitments $\mathtt{com}_{b_1}, \dots, \mathtt{com}_{b_k}$ that appear in the common first round of the two transcripts are binding and are thus opened to the same strings $a_{b_1}, \dots, a_{b_k}$ in the two transcripts. Therefore, for $i = 1, \dots, k$, we can extract from T_1 and T_2 two transcripts of Π: (a_{b_i}, c_1, z_{b_i}) for input $x_{b_i}^1$ and (a_{b_i}, c_2, z'_{b_i}) for input $x_{b_i}^2$. Each pair of transcripts gives, by the adaptive-input special soundness of Π, a witness for $x_{b_i}^1$ and one for $x_{b_i}^2$. □

□

6.4.2 Adaptive-Input Witness Indistinguishability

Here we prove that Π_k is WI with respect to a PPT adversary $\mathcal{A}$ that is allowed to select the instance and the witnesses after receiving the first round. We have the following theorem.

Theorem 18. *Under the DDH assumption, if* Π *is SHVZK for* Rel *then* Π_k *is adaptive-input WI for relation* Rel_k*.*

Proof. For sake of contradiction, let $\mathcal{A}$ be a PPT adversary and aux an auxiliary information for which $\mathsf{AdvAWI}_{\Pi_k,\mathcal{A}}(\lambda, \mathsf{aux})$ is a non-negligible function of λ. We let $X = (x_1, \dots, x_n)$ denote the instance output by $\mathcal{A}$ at Step 2 of $\mathsf{ExpAWI}_{\Pi_k,\mathcal{A}}$ and we let $W^0 = ((w_1^0, d_1^0), \dots (w_k^0, d_k^0))$ and $W^1 = ((w_1^1, d_1^1), \dots (w_k^1, d_k^1))$ denote the witnesses output. We remark that $(x_{d_i^b}, w_i^b) \in \mathsf{Rel}$ for $i = 1, \dots, k$ and $b = 0, 1$ and that $i \neq j$ implies that $d_i^0 \neq d_j^0$ and $d_i^1 \neq d_j^1$. Let $m \leq k$ be the number of instances of Π in X for which W^1 contains a witness but W^0 does not. Obviously, since W^0 and W^1 contain witnesses for the same number k of instances of Π in X, it must be the case that m is also the number of instances of Π in X for which W^0 contains a witness and W^1 does not. We call m the number of *unique* instances (these are instances for which a witness appears in exactly one of W^0 and W^1).

For integer $0 \leq m \leq k$ for which $\mathcal{A}$ has a positive probability of outputting X, W^0, W^1 with m unique instances, we define $\mathsf{AdvAWI}(\lambda, \mathsf{aux}|m)$ to be the advantage of $\mathcal{A}$ conditioned on the event that there are m unique instances. Notice that we dropped the indices $\mathcal{A}$ and Π_k not to overburden the notation. Since $\mathsf{AdvAWI}(\lambda, \mathsf{aux})$ is non-negligible in λ there must exist integer $0 \leq m \leq k$ such that $\mathsf{AdvAWI}(\lambda, \mathsf{aux}|m)$ is defined and non-negligible and the probability of having m unique instances is also non-negligible. Notice that the value of m depends solely on the adversary $\mathcal{A}$ and we consider it fixed throughout the proof. In designing the reductions to the problem of distinguishing DH tuples from 1-non-DH tuples (which, by Lemma 7, is hard under the DDH assumption), we shall assume that $\mathcal{A}$ would output X, W^0 and W^1 with m unique instances. When this is actually the case, which by our choice of m has a non-negligible probability of occurring, we will have a non-negligible advantage in solving the problem; when instead the number of unique instances is other than m, we will output a random guess. This is sufficient to prove that a non-negligible advantage in distinguishing DH tuples from 1-non-DH tuples is obtained.

We rename the instances of X output by $\mathcal{A}$, so that W^0 and W^1 can be written as

$$W^0 = \big((w_1^0, m+1), \ldots, (w_m^0, 2m), (w_{m+1}^0, 2m+1), (w_k^0, m+k)\big)$$

and

$$W^1 = \big((w_1^1, 1), \ldots, (w_m^1, m), (w_{m+1}^1, 2m+1), \ldots, (w_k^1, m+k)\big);$$

that is, we sort the instances of X so that the first m instances, $x_1, \ldots, x_m$, have a witness only in W^1; the next m instances, $x_{m+1}, \ldots, x_{2m}$, have a witness only in W^0; and the last $k-m$ instances, $x_{2m+1}, \ldots, x_{m+k}$, have witnesses in the both. For any two such W^0 and W^1, we define the following intermediate sequences of witnesses $W_1, \ldots, W_k$:

1. For $i = 0, \ldots, m$: W_i consists of witnesses

   $$W_i = \big(\left(w_1^1, 1\right), \ldots, \left(w_i^1, i\right), \left(w_{i+1}^0, m+i+1\right), \ldots, \left(w_m^0, 2m\right), \left(w_{m+1}^0, 2m+1\right), \ldots, \left(w_{m+k}^0, m+k\right)\big).$$

   Note that W_i contains witnesses for $(x_1, \ldots, x_i, x_{m+1+i}, \ldots, x_{2m})$. Moreover, W_0 coincides with W^0 and in W_m the first m witnesses are from W^1 and the remaining are from W^0.

2. For $i = m+1, \ldots, k$: W_i consists of witnesses

   $$\begin{aligned} W_i = \big((w_1^1, 1), \ldots, (w_m^1, m), (w_{m+1}^1, 2m+1), \ldots \\ \ldots, (w_{m+i}^1, m+i), (w_{m+i+1}^0, m+i+1), \ldots, (w_{m+k}^0, m+k)\big). \end{aligned}$$

   It is easy to see that W_k coincides with W^1.

For $i = 0, \ldots, k$, we define hybrid experiment $\mathcal{H}_i$ as the experiment in which the challenger $\mathcal{C}$ uses sequence of witnesses W_i to complete the third step of the experiment $\mathsf{ExpAWI}_{\Pi_k,\mathcal{A}}$. Clearly, $\mathcal{H}_0$ is the experiment $\mathsf{ExpAWI}_{\Pi_k,\mathcal{A}}$ when $\mathcal{C}$ picks $b = 0$ and $\mathcal{H}_k$ is the same experiment when $\mathcal{C}$ picks $b = 1$.

We start by proving indistinguishability of $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ for $i = 0, \ldots, m-1$. We will then argue about the indistinguishability of $\mathcal{H}_{m+i}$ and $\mathcal{H}_{m+i+1}$ for $i = 0, \ldots, k-m-1$. We assume inductively that in $\mathcal{H}_i$ the probability that there will be m unique instances is non-negligible and prove that this is still the case in $\mathcal{H}_{i+1}$. We remind the reader that, in $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$, the challenger $\mathcal{C}$ uses witnesses for the following k instances:

$\mathcal{H}_i$	$x_1 \cdots x_i$		x_{m+i+1}	$x_{m+i+2} \cdots x_{2m}$	$x_{2m+1} \cdots x_{m+k}$
$\mathcal{H}_{i+1}$	$x_1 \cdots x_i$	x_{i+1}		$x_{m+i+2} \cdots x_{2m}$	$x_{2m+1} \cdots x_{m+k}$

The differences between the two hybrid experiments are relative to instances x_{i+1} and x_{m+i+1} and to the witness used by the prover to complete the proof that at least k tuples are $1-non-DH$. Specifically,

1. the transcript of Π for x_{i+1} is produced by the simulator in $\mathcal{H}_i$ and by the prover in $\mathcal{H}_{i+1}$;
2. the first messages of the transcript of Π for x_{i+1} is committed to with a DH tuple in $\mathcal{H}_i$ and with a 1-non-DH tuple in $\mathcal{H}_{i+1}$;
3. the transcript of Π for x_{m+i+1} is produced by the prover in $\mathcal{H}_i$ and by the simulator in $\mathcal{H}_{i+1}$;
4. the first messages of the transcript of Π for x_{m+i+1} is committed to with a 1-non-DH tuple in $\mathcal{H}_i$ and with a NDH tuple in $\mathcal{H}_{i+1}$;

5. protocol Π^{1nDH} is completed by using the witnesses for the 1-non-DH tuples associated with $x_1, \cdots, x_i, x_{m+i+2}, \cdots, x_{2m}, x_{2m+1}, \cdots, x_{m+k}$. In addition, the witness for the 1-non-DH tuple associated with x_{m+i+1} is used by $\mathcal{H}_i$ and the witness for the 1-non-DH tuple associated with x_{i+1} is used by $\mathcal{H}_{i+1}$.
6. in addition, in both hybrid experiments exactly k of the tuples are 1-non-DH and exactly $n-k$ are DH.

To prove indistinguishability of $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ we consider four intermediate hybrids: $\mathcal{H}_i^1, \dots, \mathcal{H}_i^4$.

$\boldsymbol{\mathcal{H}_i^1}$ differs from $\mathcal{H}_i$ in the way the transcript of Π instance x_{i+1} is computed. Specifically, instead of using the SHVZK simulator of Π, $\mathcal{H}_i^1$ uses the algorithm of the prover of Π run with w_{i+1} as input. Indistinguishability of $\mathcal{H}_i^1$ and $\mathcal{H}_i$ follows directly from the following two observations. First, observe that x_{i+1} is associated in both $\mathcal{H}_i$ and $\mathcal{H}_i^1$ with a DH tuple and therefore the commitment is perfectly hiding. Moreover, Π's simulator is perfect.

$\boldsymbol{\mathcal{H}_i^2}$ differs from $\mathcal{H}_i^1$ in the way the tuples used to compute the commitments are chosen. Specifically, $\mathcal{H}_i^2$ chooses $k+1$ 1-non-DH tuples, $n-k-1$ DH tuples (as opposed to k 1-non-DH tuples and $n-k$ DH tuples). The extra 1-non-DH tuple is used by $\mathcal{H}_i^2$ to compute the commitment of the first message of Π that will be associated to x_{i+1} in the third round. As a sanity check, notice that in this case the commitment associated with x_{i+1} is binding but this is not a problem as an accepting transcript of Π for x_{i+1} can be computed by using witness w_{i+1}.

We observe that Lemma 7 implies that the probability that, in $\mathcal{H}_i^2$, $\mathcal{A}$ produces two sequences W^0 and W^1 of witnesses with m unique instances is still non-negligible[5].

Indistinguishability of $\mathcal{H}_i^1$ and $\mathcal{H}_i^2$ follows by Lemma 7. Let $T = (g, A, B, X)$ be either a random 1-non-DH tuple or a random DH tuple and consider the following simulator algorithm S that on input T produces the view of $\mathcal{A}$ in $\mathcal{H}_i^1$ or in $\mathcal{H}_i^2$ depending on whether T is DH or non-DH. The n tuples selected by S consist in k randomly selected 1-non-DH tuples, $n-k-1$ randomly selected DH tuples and T. Then, S executes the same steps as in $\mathcal{H}_i^1$ and will associate tuple T with x_{i+1}. Notice that S does not need a witness for T being DH to complete the execution as the accepting transcript of Π for x_{i+1} is computed using the witness w_{i+1} and the prover's algorithm and thus there is no need of a trapdoor to change the first message committed in the 1st round.

$\boldsymbol{\mathcal{H}_i^3}$ differs from $\mathcal{H}_i^2$ as protocol Π^{1nDH} is completed by using the witness of the tuple associated with x_{i+1} and not the tuple associated with x_{m+i+1} (along, obviously, the tuples associated with $x_1, \dots, x_i$ and with $x_{m+i+2}, \dots, x_{m+k}$). Indistinguishability of $\mathcal{H}_i^3$ and $\mathcal{H}_i^2$ follows from the perfect WI of Π^{1nDH}. Notice also that the only change intervenes after $\mathcal{A}$ has output the instances and the witnesses and therefore the probability of having m unique instances does not change.

$\boldsymbol{\mathcal{H}_i^4}$ differs from $\mathcal{H}_i^3$ in the way the tuples used to compute the commitments are chosen. Specifically, $\mathcal{H}_i^4$ chooses k 1-non-DH tuples, $n-k$ DH tuples (as opposed to $k+1$ 1-non-DH tuples and $n-k-1$ DH tuples). The extra DH tuple is used by $\mathcal{H}_i^4$ to compute the commitment of the first message of Π that will be associated to x_{m+i+1} in the third round. Indistinguishability of $\mathcal{H}_i^3$ and $\mathcal{H}_i^4$ follows by the same argument used for the indistinguishability of $\mathcal{H}_i^2$ and $\mathcal{H}_i^1$ and also by the same argument the probability of having m unique instances stays non-negligible.

Finally, we observe that $\mathcal{H}_i^4$ differs from $\mathcal{H}_{i+1}$ in the way the transcript of Π instance x_{m+i+1} is computed. Specifically, instead of using the prover of Π, $\mathcal{H}_i^4$ uses the simulator of Π. Indistinguishability of $\mathcal{H}_i^4$ and $\mathcal{H}_{i+1}$ follows by the same argument used for the indistinguishability of

[5]Actually, the probability is, up to a negligible additive factor, the same as in $\mathcal{H}_i^1$ which in turn is exactly the same as in $\mathcal{H}_i$

$\mathcal{H}_i$ and $\mathcal{H}_i^1$. Notice also that the only change intervenes after $\mathcal{A}$ has output the instances and the witnesses and therefore the probability of having m unique instances does not change.

We have thus proved that $\mathcal{H}_0$ is indistinguishable from $\mathcal{H}_m$. To complete the proof, we need to prove that $\mathcal{H}_{m+i}$ and $\mathcal{H}_{m+i+1}$ are indistinguishable for $i = 0, \dots, k-m-1$. This follows directly from the observation that $\mathcal{H}_{m+i}$ and $\mathcal{H}_{m+i+1}$ only differ in the witness used for x_{2m+i+1}: $\mathcal{H}_{m+i}$ uses the witness from W^0 whereas $\mathcal{H}_{m+i+1}$ uses the witness from W^1. Indistinguishability then follows directly from the Perfect WI of Π.

6.4.3 Online performances

In Table 6.2 of Section 6.1.4 we have compared the number of exponentiations made by the prover in the online phase of [LS90, CDS94, CPS+16a], with the exponentiations require by the prover of Π_k. More precisely, the third row of Table 6.2 refers to the case where Π_k is compiled by using a Σ-protocol for the logarithm discrete relation, and the last row refers to the case where an adaptive-input 2-special sound[6] protocol is used instead. Let us now show why we obtain $2(n-k)$ for the forme case, and $4(n-k)$ for the latter.

In the first case our construction involves $10n-k$ exponentiations in total, considering both the offline and the online phases. In our construction in the 1st round we sample $n-k$ DH tuples and k non-DH, sampling a DH/non-DH tuple costs 3 exponentiations, so this operation costs $3n$. Given that a commitment computed according to the commitment scheme described previously based on DH tuples costs 4 exponentiations and that in the 1st round we compute $n-k$ equivocal commitments and k binding commitments, this sums up to $2n+2k$ modular exponentiations. Furthermore the prover computes the 1st round of Schnorr's Σ-protocol n times and this costs n exponentiations. Moreover it has to run [CDS94] to prove knowledge of witnesses for k instances out of n instances, and this costs $2n-k$ exponentiations. The only operations that involve exponentiations at the third round are the $n-k$ executions of the simulator of Schnorr's Σ-protocol. Therefore the online phase costs $2(n-k)$.

The Adaptive Proof of Knowledge version of our construction costs $13n-3k$ exponentiations in total. We first observe that in the adaptive-input 2-special sound version of Schnorr's Σ-protocol an execution of the simulator costs 4 exponentiations and that computing the 1st round using just the honest prover procedure involves 2 exponentiations. Hence the first round of our Adaptive Proof of Knowledge construction involves $6n+k$ exponentiations and the online phase costs $4(n-k)$ exponentiations. The analysis for the case of 1 out of 2 is similar with $k=1$ and $n=2$ but in this case, in the offline phase, we do not consider the cost of [CDS94] since the correctness of the pair of tuples can be self-verified.

$\square$

6.5 Extension to Multiple Relations

In this section, we generalize the result of Section 6.4 to the case of different relations. More specifically, given delayed-input Σ-protocols $\Sigma_1, \dots, \Sigma_n$ for polynomial-time relations $\mathsf{Rel}_1, \dots, \mathsf{Rel}_n$, we construct an Adaptive-Input Proof of Partial Knowledge $\Gamma_k^{\mathsf{Rel}_1,\dots,\mathsf{Rel}_n} = (\mathcal{P}_{\Gamma_k}{}^{\mathsf{Rel}_1,\dots,\mathsf{Rel}_n}, \mathcal{V}_{\Gamma_k}^{\mathsf{Rel}_1,\dots,\mathsf{Rel}_n})$

[6]We recall that by Theorem 17, when an adaptive-input 2-special sound protocol is used as input of our compiler we obtain that Π_k is Adaptive Proof of Knowledge.

for the k-threshold polynomial-time relation

$$\mathsf{Rel}^k(\mathsf{Rel}_1,\dots,\mathsf{Rel}_n) = \Big\{\Big(\big(x_1,\dots,x_n\big),\big((w_1,d_1)\dots,(w_k,d_k)\big)\Big) : 1 \le d_1 < \dots < d_k \le n \\ \text{and } (x_{d_i},w_i) \in \mathsf{Rel}_{d_i} \text{ for } i=1,\dots,k\Big\}.$$

Not to overburden our notation, we will just write $\mathsf{Rel}^k, \Gamma_k, \mathcal{P}_{\Gamma_k}$, and $\mathcal{V}_{\Gamma_k}$, whenever $\mathsf{Rel}_1,\dots,\mathsf{Rel}_n$ are clear from the context.

Let us start by providing some intuition on Γ_k. For $j=1,\dots,n$, the prover of Γ_k computes two first-round messages for Σ_j and commits to each one using a different tuple. Note that Σ_j is delayed-input so this step can be performed without knowing the instances (nor the witnesses). The two tuples, T_j^0 and T_j^1, used to compute the commitments for j are chosen so that they share the first three components and T_j^1 is DH and, consequently, T_j^0 is non-DH. Therefore, the verifier is guaranteed that, for each j, at most one of the commitments of the tuple is DH (and thus at most one commitment can be equivocated). For each j, the tuples T_j^0 and T_j^1 and the relative commitments $\mathtt{com}_j^0$ and $\mathtt{com}_j^1$ are sent in random order to the verifier. The prover then receives the n instances, witnesses for k of them and the verifier's challenge. Then, for each instance x_j for which a witness is available, the prover decommits the commitment computed using T_j^0 (the non-DH tuple) and computes the third round message by running the prover of Σ_j. On the other hand, for each instance x_j for which no witness is provided, the prover runs the simulator of Σ_j and produces an accepting transcript; then the commitment computed using T_j^1 (the DH tuple) is opened as the first message of the simulated transcript, by using equivocation. In both cases, the decommitment and the accepting transcript are sent to the verifier.

The verifier expects to receive an accepting transcript for each instance with the first message coming (through equivocation for $n-k$ of them) from a commitment sent as part of the first round. Moreover, to ensure that the prover has not cheated by equivocating too many commitments, the prover and the verifier engage in parallel in a protocol by which the prover proves that from among the n unopened commitments at least k are associated with DH tuples; this implies, by the way the pairs of tuples have been constructed, that at least k of the opened commitments are associated with non-DH tuples. We denote the resulting protocol by $\mathsf{ThresDDH} = (\mathcal{P}_{\mathsf{TDDH}}, \mathcal{V}_{\mathsf{TDDH}})$. Notice that when $\mathsf{ThresDDH}$ starts, the prover does not know which commitment will be opened (since this depends on the witnesses received after the challenge); this can be handled by our protocol described in Sec. 6.4, since it only needs the instances and witnesses before the third round starts.

More formally, we use the following tools to construct Γ_k.

1. We use as a commitment scheme an IBTC for the polynomial time relation $\mathsf{DH} = \{((g,A,B,X),\alpha) : A = g^\alpha \text{ and } X = B^\alpha\}$.

2. A delayed-input, adaptive-input 2-special sound, adaptive-input WI protocol for the relation

$$\mathsf{Rel}_{\mathsf{TDDH}} = \Big\{\Big(\big(T_1,\dots,T_n\big),\big((\alpha_1,d_1),\dots,(\alpha_k,d_k)\big)\Big) : 1 \le d_1 < \dots < d_k \le n \\ \text{and } (T_{d_i},\alpha_i) \in \mathsf{DH}, \text{ for } i=1,\dots,k\Big\}.$$

   We observe that this protocol can be obtained relying on Theorem 18, and by combining the fact that the well known Σ-protocol for DH tuple [DH76] can be converted into one that enjoys adaptive-input 2-special soundness (see. Sec. 6.3.2) with Lemma 6.4.1.

1st round. $\mathcal{P}_{\Gamma_k} \Rightarrow \mathcal{V}_{\Gamma_k}$:

$\mathcal{P}_{\Gamma_k}$ receives as unary inputs the security parameter λ, the number n of theorems that will be given as input at the beginning of the third round, and the number k of witnesses that will be provided.

1. Set $(\mathcal{G}, p, g) \leftarrow \mathsf{GG}(1^\lambda)$.
2. For $j = 1, \ldots, n$
   - 2.1. Randomly sample a non-DH tuple $T_j^0 = (g_j, A_j, B_j, X_j)$ over $\mathcal{G}$, along with α_j such that $A_j = g_j^{\alpha_j}$.
   - 2.2. Set $Y_j = B_j^{\alpha_j}$ and $T_j^1 = (g_j, A_j, B_j, Y_j)$ (note that, by construction, T_j^1 is a DH tuple).
3. Select random string R and use it to compute the first round message a^{ThresDDH} of $\mathsf{ThresDDH}$ by running prover $\mathcal{P}_{\mathsf{TDDH}}$.
   Send a^{ThresDDH} to $\mathcal{V}_{\Gamma_k}$.
4. For $j = 1, \ldots, n$
   - 4.1. Select random strings R_j^0 and R_j^1 and use them to compute the first-round messages a_j^0 and a_j^1 of Σ_j by running the prover of Σ_j.
   - 4.2. Compute the pair $(\mathtt{com}_j^0, \mathtt{dec}_j^0)$ of commitment and decommitment of the message a_j^0 using non-DH tuple T_j^0.
   - 4.3. Compute the commitment $\mathtt{com}_j^1$ (of the message a_j^1) using the DH tuple T_j^1.
   - 4.4. Send pairs $(T_j^0, \mathtt{com}_j^0)$ and $(T_j^1, \mathtt{com}_j^1)$ in random order to $\mathcal{V}_{\Gamma_k}$.

2nd round. $\mathcal{V}_{\Gamma_k} \Rightarrow \mathcal{P}_{\Gamma_k}$: $\mathcal{V}_{\Gamma_k}$ randomly selects a challenge c and sends it to $\mathcal{P}_{\Gamma_k}$.

3rd round. $\mathcal{P}_{\Gamma_k} \Rightarrow \mathcal{V}_{\Gamma_k}$:

$\mathcal{P}_{\Gamma_k}$ receives theorems $x_1, \ldots, x_n$ and, for $d_1 < \ldots < d_k$, witnesses $w_1, \ldots, w_k$ for theorems $x_{d_1}, \ldots, x_{d_k}$, respectively. We let $\widetilde{d}_1 < \ldots < \widetilde{d}_{n-k}$ denote the indices of the theorems for which no witness has been provided.

1. For $l = 1, \ldots, k$
   - 1.1. Use j as a shorthand for d_l.
   - 1.2. Set $U_j = T_j^1$ and $\hat{U}_j = T_j^0$.
   - 1.3. Compute z_j by running prover of Σ_j on input (x_j, w_l), randomness R_j^0 used to compute the first round a_j^0, and challenge c.
   - 1.4. Set $M_j = (a_j^0, z_j, \mathtt{dec}_j^0, \hat{U}_j)$.
2. For $l = 1, \ldots, n-k$
   - 2.1. Use j as a shorthand for $\widetilde{d}_l$.
   - 2.2. Set $U_j = T_j^0$ and $\hat{U}_j = T_j^1$.
   - 2.3. Run the simulator of Σ_j on input x_j and c therefore obtaining $(\widetilde{a}_j^1, z_j)$.
   - 2.4. Use the trapdoor α_j to compute the decommitment $\mathtt{dec}_j^1$ of $\mathtt{com}_j^1$ as $\widetilde{a}_j^1$.
   - 2.5. Set $M_j = (\widetilde{a}_j^1, z_j, \mathtt{dec}_j^1, \hat{U}_j)$.
3. For $l = 1, \ldots, n$ send M_l to $\mathcal{V}_{\Gamma_k}$.

4. Compute the third round z^{ThresDDH} of $\mathsf{ThresDDH}$ by running $\mathcal{P}_{\mathsf{TDDH}}$ on input tuples $(U_1, \ldots, U_n)$, witnesses $\alpha_{d_1}, \ldots, \alpha_{d_k}$ and randomness R used to compute the first round a^{ThresDDH}.

$\mathcal{V}_{\Gamma_k}$ accepts if and only if the following conditions are satisfied.

1. For $j = 1, \ldots, n$
   Check that the two tuples sent for j in the first round share the first three components. Write M_j as $M_j = (a_j, z_j, \mathtt{dec}_j, \hat{U}_j)$.
   Check that (a_j, c, z_j) is an accepting conversation of Σ_j for instance x_j.
   Check that $\mathtt{dec}_j$ is a decommitment as a_j with respect to tuple $\hat{U}_j$ of one of $\mathtt{com}_j^0$ and $\mathtt{com}_j^1$ and that tuple $\hat{U}_j$ was associated with j in the first round. Denote by U_j the other tuple associated with j.
2. Check that $(a^{\mathsf{ThresDDH}}, c, z^{\mathsf{ThresDDH}})$ is an accepting conversation of $\mathcal{V}_{\mathsf{TDDH}}$ for instances $U_1, \ldots, U_n$.

6.5.1 (Adaptive-Input) Proof of Knowledge

In this section, we study the PoK properties of Γ_k. We start by proving in Theorem 19 that Γ_k is a PoK. Then, in Theorem 20, we prove that, under the additional assumption that $\Sigma_1, \ldots, \Sigma_n$ are adaptive-input 2-special sound, Γ_k is an adaptive-input PoK.

Theorem 19. *Γ_k is a proof of knowledge for $\mathsf{Rel}^k(\mathsf{Rel}_1, \ldots, \mathsf{Rel}_n)$ with knowledge error negligible in the length of the challenge.*

Proof. The completeness property follows from the completeness of protocols $\mathsf{ThresDDH}$ and $\Sigma_1, \ldots, \Sigma_n$, and from the correctness and the trapdoor property of the Instance-Dependent Binding Commitment scheme used. We next prove that Γ_k is t-special sound, with $t = n-k+2$. The theorem then follows by Theorem 9.

To prove that Γ_k is t-special sound we describe an algorithm that takes t accepting transcripts of Γ_k for the same input $(x_1, \ldots, x_n)$ that share the first message and outputs a witness for $(x_1, \ldots, x_n)$. We remind the reader that the first message contains n binding commitments to the first message of each of protocols $\Sigma_1, \ldots, \Sigma_n$. The last message of each transcript instead contains the openings of k of the binding commitments[7].

The algorithm examines the t transcripts $\tau^1, \ldots, \tau^t$ one at the time looking for a revealing transcript. We say that transcript τ^j is *revealing* if each of the k binding commitments opened in the last message of τ^j has been opened at least once in one of the preceding $j-1$ transcripts. Once the algorithm has found a revealing transcript, it can obtain a collision for k of the n Σ-protocols $\Sigma_1, \ldots, \Sigma_n$ from which, by special soundness, it obtains the witness for k of the inputs of transcript τ^j. We observe that the special soundness of $\Sigma_1, \ldots, \Sigma_n$ is sufficient to extract k witness since in τ^j every accepting transcript for Σ_i is with respect to the same theorem x_i for all $i \in \{1, \ldots, n\}$.

We conclude the proof by showing that any set of $t = n - k + 2$ transcripts contains at least one revealing transcript. Clearly, the k binding commitments opened in the last message of the first transcript are all opened for the first time. In each subsequent transcript that is not revealing there exists at least one binding commitment that is opened for the first time. Therefore, if, after $n-k+1 = t-1$ transcripts have been examined, no revealing transcript has been found then the t-th transcript must be a revealing transcript. □

[7]Since $t > 1$ and $\mathsf{ThresDDH}$ is adaptive-input 2-special sound, then we can claim that at least k of the opened commitments in the last round are binding.

We now look at adaptive-input PoK.

Theorem 20. *If $\Sigma_1, \dots, \Sigma_n$ enjoy adaptive-input 2-special soundness then Γ_k is an adaptive-input proof of knowledge for $\mathsf{Rel}^k(\mathsf{Rel}_1, \dots, \mathsf{Rel}_n)$ with knowledge error negligible in the length of the challenge.*

Proof. The delayed-input completeness of Γ_k follows from the delayed-input completeness of protocol $\mathsf{ThresDDH}$, the correctness and trapdoor property of the Instance-Dependent Binding Commitment scheme used and from the delayed-input completeness of Σ_i for $i = 1, \dots, n$. In order to prove that Γ_k enjoys the property of adaptive-input PoK we need to show an extractor $\mathsf{AExtract}$ that, given oracle access to $\mathcal{P}_{\Gamma_k}{}^{\star}$, outputs a couple (x, w) according to Definition 25. Before that, we prove the following lemma.

Lemma 9. *There exists an algorithm that, on input $m = n - k + 2$ accepting transcripts of protocol Γ_k with the same first message and distinct challenges, outputs in time polynomial in λ a witness for the input of one of the m transcripts.*

Proof. This proof follows the same approach used in the proof of Theorem 19. The main difference is that in this case we rely on the adaptive-input 2-special soundness instead of the special-soundness of $\Sigma_1, \dots, \Sigma_n$ to extract the witnesses. The reason why special-soundness is not sufficient in this case is due to the additional power given to the malicious prover to choose different inputs for different transcripts.

We remind the reader that the m accepting transcripts share the first message that contains n binding commitments to the first messages of protocols $\Sigma_1, \dots, \Sigma_n$. The last message of each transcript instead contains the openings of k of the binding commitments. The algorithm examines the m transcripts $\tau^1, \dots, \tau^m$ one at the time looking for a revealing transcript. Once the algorithm has found a revealing transcript, it can obtain a collision for k of the n Σ-protocol $\Sigma_1, \dots, \Sigma_n$ from which, by adaptive-input 2-special soundness, it obtains the witness for k of the inputs of transcript τ^j. We conclude the proof by showing that any set of m transcripts contains at least one revealing transcript. Clearly, the k binding commitments opened in the last message of the first transcript are all opened for the first time. In each subsequent transcript that is not revealing there exists at least one binding commitment that is opened for the first time. Therefore, if, after $n - k + 1 = m - 1$ transcripts have been examined, no revealing transcript has been found then the m-th transcript must be a revealing transcript.

□

Now we are ready to show how our extractor $\mathsf{AExtract}$ works when given access to malicious prover $\mathcal{P}^{\star}$.

1. Upon receiving a from $\mathcal{P}^{\star}$, $\mathsf{AExtract}$ randomly selects challenge $c^1 \leftarrow \{0,1\}^{\lambda}$ and sends it to $\mathcal{P}^{\star}$.

2. Upon receiving z^1 and the theorem x^1, $\mathsf{AExtract}$ runs $\mathcal{V}_{\Gamma_k}$ to check if (a, c^1, z^1) is accepting with respect to x^1. If $\mathcal{V}_{\Gamma_k}$ rejects, $\mathsf{AExtract}$ outputs $\perp$ and stops, otherwise it continues as follows.

3. For $i = 2, \dots, t$

   $\mathcal{P}^{\star}$ rewinds $\mathcal{P}^{\star}$ by sending a new randomly selected challenge $c_i \leftarrow \{0,1\}^{\lambda}$ different from all previous challenges. We denote by (a, c^i, z^i) the transcript obtained and by x^i the relative input.

4. If t accepting transcripts have been computed then AExtract continues; otherwise, it stops and outputs $\perp$.

5. Relying on Lemma 9, AExtract uses the t accepting conversations to compute and output a witness for one of the inputs $x^1, \ldots, x^t$.

We now analyze the running time of AExtract. Let $p^\star(\lambda)$ be the probability that $\mathcal{P}^\star$ provides an accepting transcript for a challenge of length λ. We next show that if $p^\star > (n-k+1)2^{-\lambda}$ then AExtract is an extractor for Γ_k. The probability that in the step 3, for a randomly selected challenge c_i, with $c_1 \neq \cdots \neq c_{i-1} \neq c_i$, $\mathcal{P}^\star$ provides an accepting transcript is $p^\star(\lambda) - (i-1)2^{-\lambda}$. Therefore the expected number of rewinds that have to be made in step 3, for some $i \in \{2, \ldots, t\}$ is $\frac{1}{p^\star(\lambda)-(i-1)2^{-\lambda}}$. This means that the expected number of steps made by $\mathsf{AExtract}^{\mathcal{P}^\star}$ is

$$\sum_{i=2}^{t} \frac{1}{p^\star(\lambda)-(i-1)2^{-\lambda}} < \frac{t-1}{p^\star(\lambda)-(t-1)2^{-\lambda}}.$$

□

6.5.2 Adaptive-Input Witness Indistinguishability

Theorem 21. *If relations* $\mathsf{Rel}_1, \ldots, \mathsf{Rel}_n$ *admit a SHVZK* Σ*-protocol then, under the DDH assumption,* Γ_k *is adaptive-input WI for* $\mathsf{Rel}^k(\mathsf{Rel}_1, \ldots, \mathsf{Rel}_n)$*.*

Proof. We adopt the same framework of the proof for the case of one relation. Specifically, we have the same definition of hybrid witness sequence W_i and of hybrid experiment $\mathcal{H}_i$, for $i = 0, \ldots, k$. We start by proving indistinguishability of $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ for $i = 0, \ldots, m-1$. The differences between the two hybrids are relative to inputs x_{i+1} and x_{m+i+1} and the witness used to complete protocol ThresDDH. We consider the following intermediate hybrids.
$\boldsymbol{\mathcal{H}_i^1}$ differs from $\mathcal{H}_i$ in the way the transcript of Σ_{i+1} for instance x_{i+1} is computed. Specifically, instead of using the SHVZK simulator of Σ_{i+1}, $\mathcal{H}_i^1$ uses the algorithm of the prover of Σ_{i+1} run with w_{i+1} as input. Notice that in both hybrids the tuple $\widehat{U}_{i+1}$ used to commit the first-round message of Σ_{i+1} is a DH tuple whereas U_{i+1} is a non-DH tuple. In $\mathcal{H}_i^1$ however the equivocability of the commitment is not used and the transcript is completed using the witness and the prover's algorithm for Σ_{i+1}. Indistinguishability of $\mathcal{H}_i^1$ and $\mathcal{H}_i$ follows directly from the perfect SHVZK.
$\boldsymbol{\mathcal{H}_i^2}$ differs from $\mathcal{H}_i^1$ in the fact that tuple $\widehat{U}_{i+1}$ is a non-DH tuple and U_{i+1} is a DH tuple. In other words, in $\mathcal{H}_i^2$ the first-round message of the transcript relative to x_{i+1} shown at the third round has been committed to by a non-DH tuple.

Suppose now, for sake of contradiction, that there exists a PPT distinguisher D and a polynomial $\mathsf{poly}(\cdot)$ such that, for sufficiently large security parameters λ,

$$p_1(\lambda) \geq p_2(\lambda) + 1/\mathsf{poly}(\lambda),$$

where $p_1(\lambda)$ and $p_2(\lambda)$ are, respectively, the probabilities that D outputs 1 on input the view of $\mathcal{A}$ in $\mathcal{H}_i^1$ and in $\mathcal{H}_i^2$. We use D and $\mathcal{A}$ to design an algorithm $\mathcal{B}$ that has a non-negligible advantage in the DDH game. Algorithm $\mathcal{B}$ receives as input a challenge tuple $T = (g, A, B, X)$, randomly selects Y and constructs the tuple $\widehat{T} = (g, A, B, Y)$ that shares the first three components with T. Then $\mathcal{B}$ simulates the view of $\mathcal{A}$ in $\mathcal{H}_1^i$ with the following two modifications: T and $\widehat{T}$ are the tuples associated with $i+1$; in the third round, $\mathcal{B}$ selects one of the two at random and the commitment computed at the first round with respect to the selected tuple is opened. $\mathcal{B}$ feeds D with the view generated by interacting with $\mathcal{A}$ and records D's output. Finally, $\mathcal{B}$ outputs 1 if and only if it had selected T and D has output 1 or it had selected $\widehat{T}$ and D has output 0.

Now, let us compute the probability that $\mathcal{B}$ outputs 1 when T is DH tuple. Notice that if T is selected (and this happens with probability 1/2) then $\mathcal{B}$ has produced the view of $\mathcal{H}_i^1$ and thus the probability that D outputs 1 is p_1. On the other hand, if $\widehat{T}$ is selected then $\mathcal{B}$ has produce the view of $\mathcal{H}_i^2$ and thus the probability that D outputs 0 is $1 - p_2$. Therefore, it T is DDH, $\mathcal{B}$ outputs 1 with probability $1/2 \cdot (1 + p_1 - p_2)$.

Consider now the case in which T is non-DH. In this case, both T and $\widehat{T}$ are non-DH and thus the view received by D is independent from which one of the two is selected by $\mathcal{B}$. We thus denote by p the probability that D outputs 1 when the view contains two non-DH tuples. As in the previous case, $\mathcal{B}$ outputs 1 if T is selected and D outputs 1 (this event has probability $1/2p$) and $\widehat{T}$ is selected and D outputs 0 (this event has probability $1/2 \cdot (1 - p)$). Therefore $\mathcal{B}$ has probability 1/2 of outputting 1 when it receives a non-DH tuple in output.

We thus conclude that $\mathcal{B}$ breaks the DDH assumption. Contradiction.

$\boldsymbol{\mathcal{H}_i^3}$ differs from $\mathcal{H}_i^2$ in the witness used to compute an accepting transcript for $\mathsf{ThresDDH}$. More specifically α_{i+1} is used instead of α_{m+i+1} in the tuple that define the witness for $\mathsf{ThresDDH}$. Observe that this is possible because U_{i+1} is a DH tuple. Suppose now, for sake of contradiction, that there exists a PPT distinguisher D and a polynomial $\mathsf{poly}(\cdot)$ such that, for sufficiently large security parameters λ,

$$p_1(\lambda) \geq p_2(\lambda) + 1/\mathsf{poly}(\lambda),$$

where $p_1(\lambda)$ and $p_2(\lambda)$ are, respectively, the probabilities that D outputs 1 on input the view of $\mathcal{A}$ in $\mathcal{H}_i^2$ and in $\mathcal{H}_i^3$. We use D and $\mathcal{A}$ to design an algorithm $\mathcal{B}$ that has a non-negligible advantage in break the adaptive WI property of $\mathsf{ThresDDH}$. Algorithm $\mathcal{B}$ receives as input the first round challenge a^{ThresDDH} and computes all the other informations needed to compute the first round of the protocol of Γ_k. $\mathcal{B}$, upon receiving the challenge c computes the challenge theorem and witnesses as following:

$$X_{\mathsf{ThresDDH}} = (U_1, \dots, U_n);$$

$$W^0_{\mathsf{ThresDDH}} = (\alpha_1, \dots, \alpha_i, \alpha_{m+i+1}, \alpha_{m+i+2}, \dots, \alpha_{2m}, \alpha_{2m+1}, \dots \alpha_{m+k});$$

$$W^1_{\mathsf{ThresDDH}} = (\alpha_1, \dots, \alpha_{i+1}, \alpha_{m+i+2}, \dots, \alpha_{2m}, \alpha_{2m+1}, \dots \alpha_{m+k}).$$

Then sends them, with the challenge c, to the challenger. $\mathcal{B}$, upon receiving z^{ThresDDH}, completes the third round protocol using z^{ThresDDH}, and sends it to $\mathcal{A}$. $\mathcal{B}$ feeds D with the view generated by interacting with $\mathcal{A}$ and records D's output. Finally, $\mathcal{B}$ outputs 1 if and only if the witness W^0_{ThresDDH} has been used by the challenger and D has output 1 or has been used W^1_{ThresDDH} and D has output 0.

$\boldsymbol{\mathcal{H}_i^4}$ differs from $\mathcal{H}_i^3$ in the fact that tuple $\widehat{U}_{m+i+1}$ is a DH tuple and U_{m+i+1} is a non-DH tuple. In other words, in $\mathcal{H}_i^4$ the first-round message of the transcript relative to x_{m+i+1} shown at the third round has been committed to by a DH tuple. The indistinguishability between $\mathcal{H}_i^3$ and $\mathcal{H}_i^4$ follows the same arguments of the indistinguishability between $\mathcal{H}_i^2$ and $\mathcal{H}_i^1$

Finally, we observe that $\mathcal{H}_i^4$ differs from $\mathcal{H}_{i+1}$ in the way the transcript of Π_{m+i+1} for the instance x_{m+i+1} is computed. Specifically, instead of using the prover of Π, $\mathcal{H}_i^4$ uses the simulator of Π. Indistinguishability of $\mathcal{H}_i^4$ and $\mathcal{H}_{i+1}$ follows by the same argument used for the indistinguishability of $\mathcal{H}_i$ and $\mathcal{H}_i^1$.

We have thus proved that $\mathcal{H}_0$ is indistinguishable from $\mathcal{H}_m$. To complete the proof, we need to prove that $\mathcal{H}_{m+i}$ and $\mathcal{H}_{m+i+1}$ are indistinguishable for $i = 0, \dots, k - m - 1$. This follows directly from the observation that $\mathcal{H}_{m+i}$ and $\mathcal{H}_{m+i+1}$ only differ in the witness used for x_{2m+i+1}: $\mathcal{H}_{m+i}$ uses the witness from W^0 whereas $\mathcal{H}_{m+i+1}$ uses the witness from W^1. Indistinguishability then follows directly from the Perfect WI of Π.

□

Chapter 7

Non-Interactive Zero-Knowledge Without Programmable Random Oracles

7.1 Introduction

Non-interactive zero-knowledge (NIZK) proofs introduced in [DMP87, BFM88, BDMP91] are widely used in Cryptography. Such proofs allow a prover to convince a verifier with just one message about the membership of an instance x in a language L without leaking any additional information. NIZK proofs are not possible without a setup assumption and the one proposed initially in [BDMP91] is the existence of a *Common Reference String* (CRS) received as input both by the prover and the verifier. The CRS model has been so far the standard setup for NIZK. Another setup that has been proposed in literature is the existence of registered public keys in [BCNP04, DFN06, VV09, CG15].

Starting with the breakthrough of [FLS90, FLS99] we know that NIZK proofs in the CRS model exist for any $\mathcal{NP}$ language with the additional appealing feature of using just one CRS for any polynomial number of proofs. Moreover NIZK proofs and their stronger variations [Sah99, SCO$^+$01, GOS06] have been shown to be not only interesting for their original goal of being a non-interactive version of classic zero-knowledge (ZK) proofs [GMR85, GMR89], but also because they are powerful building blocks in many applications (e.g., for CCA encryption [NY90], ZAPs [DN00, DN07]).

Efficient NIZK. Generic constructions of NIZK proofs are rather inefficient since they require to first compute an $\mathcal{NP}$ reduction and then to apply the NIZK proof for a given $\mathcal{NP}$-complete language to the instance given in output by the reduction. A significant progress in efficiency has been proposed in [GS08] where several techniques have been proposed to obtain efficient NIZK proofs that can be used in bilinear groups.

The most popular use of NIZK proofs in real-world scenarios consists in taking an efficient *interactive* constant-round public-coin honest-verifier zero-knowledge (HVZK) proof system and in making it a NIZK argument through the so called *Fiat-Shamir (FS) transform* [FS86]. The FS transform replaces the verifier by calls to a hash function on input the transcript so far. In the random oracle [BR93] (RO) model the hash function can only be evaluated through calls to an oracle that answers as a random function. The security proof allows the simulator for HVZK to program the RO (i.e., the simulator decides how to answer to a query) and this allows to convert the entire transcript of a public-coin HVZK proof into a single message that is indistinguishable from the single message computed by a honest NIZK prover. The efficiency of the FS transform led to many practical applications. The transform is also a method to obtain signatures of knowledge, as discussed in [CL06].

In [Gro04] Groth showed an efficient transform for NIZK where soundness is proved requiring a programmable RO while no random oracle is needed to prove zero knowledge.

The risks of the FS transform. The main disadvantage of the FS transform is the fact that the random oracle methodology has been proved to be unsound both in general [CGH98] and for the specific case [GK03, BDSG$^+$13] of turning identification schemes into signatures as considered in [FS86]. Nevertheless, the examples of constructions proved secure in the RO model and insecure for any concrete hash function are seemingly artificial. Interestingly in [GOSV14] it is shown that the FS transform can be used to obtain (non-artificial) information-theoretic NIZK arguments that are not sound when knowledge of the description of the hash functions is used by the adversarial prover.

A slight modification of the FS transform gives as input to the hash function only the first round of a three-round protocol, without the instance to be proved. Despite the fact that this approach, called *weak FS transform*, has been used is literature, [BPW12] showed the insecurity of the transform when the some HVZK protocols are used (similar issues have been discussed in [CPS$^+$16b, CPS$^+$16a] in the standard model). Other weaknesses about the non-malleability of the FS transform are discussed in [FKMV12]. In contrast, there are some recent positive results [KRR17, MV16] based on obfuscation.

The FS transform applied to 3-round HVZK proofs is still one of the major uses of the RO model for real-world protocols, therefore any progress in this research direction (either on the security of the transform, or on its efficiency, or on its generality) is of extreme interest.

Efficient NIZK with designated/registered verifiers. A first attempt to get efficient NIZK arguments from some restricted class of 3-round public-coin HVZK proofs without ROs was done by [DFN06] (the proof of soundness required complexity leveraging) and later on by [VV09, CG15] that achieved a weaker form of soundness in the registered public-key model. The limitation of this model is that a NIZK proof can be verified only by a designated verifier (i.e., the proof requires a secret known to the verifier). Moreover there is an inconvenient preliminary registration phase where the verifier has to register her public key.

Lindell's transform. Very recently, in [Lin15], Lindell proposed a very interesting transform that can be seen as an attempt towards obtaining efficient constructions without random oracles. Starting from a Σ-protocol for a language L (i.e., a special type of 3-round public-coin HVZK proof used already in several efficient constructions of zero knowledge [Dam00, MP03, DCV05, Vis06, CDV06, YZ07, ABB$^+$10, OPV10, SV12]), Lindell shows how to construct an efficient NIZK[1] argument system for L in the CRS model. Two are the major advantages of Lindell's transform with respect to the FS transform. First, in Lindell's transform the proof of ZK does not need the existence of a random oracle and this allows to avoid some issues due to protocol composition [Wee09]. We remark that the proof of ZK for Lindell's transform needs a CRS but this is unavoidable as one-round ZK in the plain model is possible only for trivial languages. Second, the soundness of Lindell's transform can be proved by relying on a *non-programmable random oracle* (NPRO). An NPRO is a RO that in the protocol and in the security proofs can be used only as a black box and can not be programmed by a simulator or by the adversary of a reduction. This is a considerable advantage compared to the FS transform since replacing

[1]Lindell's NIZK argument is a not an argument of knowledge in contrast to the NIZK argument obtained through an FS transform.

a RO by an NPRO is a step towards removing completely the need of ROs in a cryptographic construction. Indeed the work of Lindell goes precisely in the direction of solving a major open problem in Cryptography: obtaining an efficient RO-free transform for NIZK arguments to be used in place of the FS transform.

The main drawback of Lindell's transform is that it requires extra computation on top of the one needed to run the Σ-protocol for the language L. In contrast, the FS transform does not incur into any overhead on top of a 3-round public-coin HVZK proof for L. In addition, since 3-round public-coin HVZK proofs are potentially less demanding than Σ-protocols, we have that requiring a Σ-protocol as starting protocol for a transform instead of a public-coin HVZK proof may already result in an efficiency loss.

Lindell's transform is based on a primitive named *dual-mode* (DM) commitment scheme (DMCS). A DMCS is based on a membership-hard language Λ and each specific commitment takes as input an instance ρ of Λ and has the following property: if $\rho \notin \Lambda$, the DM commitment is perfectly binding; on the other hand, if $\rho \in \Lambda$, the DM commitment can be arbitrarily equivocated if a witness for $\rho \in \Lambda$ is known. Moreover, the two modes are indistinguishable[2]. Lindell showed that DMCSs can be constructed efficiently from Σ-protocols for membership-hard languages and also provided a concrete example based on the language of Diffie-Hellman tuples (DH). Then, Lindell's transform shows how to combine DM commitments and Σ-protocols along with a hash function[3] to obtain an efficient NIZK argument.

7.1.1 Our Results

In this work, we continue the study of generic and efficient transforms from 3-round public-coin HVZK proofs to NIZK arguments.

We start by studying the generality and efficiency of Lindell's transform in terms of the Σ-protocol used for instantiating the DMCS (and in turn instantiating the CRS) and the Σ-protocol to which the transform is applied. As a result, we point out a significant gap in generality and efficiency of Lindell's transform compared to the FS transform.

Then we show an improved transform that is based on weaker requirements. Specifically, our transform only requires computational HVZK and optimal soundness instead of perfect special HVZK and special soundness. More interestingly and surprisingly despite being based on weaker requirements, our transform is also significantly more efficient than Lindell's transform and very close to the efficiency of the FS transform. We next discuss our contributions in more details.

The classes of Σ-protocols needed in [Lin15]. Lindell defines Σ-protocols as 3-round public-coin proofs that enjoy *perfect* special HVZK and special soundness. The former property means that the simulator on input any valid statement x and challenge e can compute (a, z) such that the triple (a, e, z) is perfectly indistinguishable from an accepting transcript where the verifier sends e as challenge. Special soundness instead means that from any two accepting transcripts (a, e, z) and (a, e', z') for the same statement x that share the first message but have different challenges $e \neq e'$, one can efficiently compute a witness w for $x \in L$. Lindell in [Lin14] shows a construction of a DMCS from any (defined as above) Σ-protocol for a membership-hard language.

The efficiency of Lindell's transform. Lindell's transform uses a DMCS derived from a Σ-protocol $\Pi_\Lambda = (\mathcal{P}_\Lambda, \mathcal{V}_\Lambda)$ for language Λ whose commitment algorithm `com` works by running the simulator of Π_Λ. The CRS contains an instance ρ of Λ along with the description of a

[2]A similar notion was introduced in [CV05, CV07] and a scheme with similar features was proposed in [DG03].
[3]In the proof of soundness this function will be modeled as an NPRO.

hash function h. The argument produced by the NIZK $\Pi = (\mathcal{P}, \mathcal{V})$ for $x \in L$ starting from a Σ-protocol $\Pi_L = (\mathcal{P}_L, \mathcal{V}_L)$ for L is computed as a tuple (a', e, z, r) where $a' = \mathtt{com}(a, r)$, $e = h(x|a')$, and z is the 3rd round of Π_L answering to the challenge e and having a as first round. The verifier checks that a' is a commitment of a with randomness r, that e is the output of $h(x|a')$ and that (a, e, z) is accepted by $\mathcal{V}_L$.

As an example, in [Lin15] Lindell discussed the use of the Σ-protocol for the language DH for which the transform produces a very efficient NIZK proof; indeed the additional cost is of only 8 modular exponentiations: 4 to be executed by the prover and 4 by the verifier.

In this work we notice however that there is a caveat when analyzing the efficiency of Lindell's transform. The caveat is due to the message space of the DMCS. Indeed, once the CRS is fixed the max length of a message that can be committed to with only one execution of $\mathtt{com}$ is limited to the challenge length l_Λ of Π_Λ. Therefore in case the first round a of Π_L is much longer than l_Λ, the transform of Lindell requires multiple executions of $\mathtt{com}$ therefore suffering of a clear efficiency loss[4].

We show indeed in Tables 7.2 and 7.3 that Lindell's transform can generate in the resulting NIZK argument a blow up of the computations compared to what $\mathcal{P}_L$ and $\mathcal{V}_L$ actually do, and therefore compared to the FS transform.

Our Transform

In this chapter, we present a different transform that is closer to the FS transform both on generality and on efficiency.

Our transform can be used to obtain a NIZK for any language L with a 3-round HVZK proofs enjoying optimal soundness (i.e., a weaker soundness requirement compared to special soundness). The CRS can be instantiated based on any membership-hard language Λ with a 3-round HVZK proofs enjoying optimal soundness. More specifically, we do not require perfect HVZK nor special HVZK for the involved Σ-protocols. Moreover, instead of special soundness, we will just require that, for any false statement and any first round message a, there is at most one challenge c that can be answered correctly. This is clearly a weaker requirement than special soundness and was already used by [MP03].

Essentially we just need that both protocols Π_L and Π_Λ are 3-round public-coin HVZK proofs with optimal soundness. Our transform produces a NIZK argument $\Pi = (\mathcal{P}, \mathcal{V})$ that does not require multiple executions of Π_L and Π_Λ and, therefore, it remains efficient under any scenario without suffering of the previously discussed issue about challenge spaces in Lindell's transform.

Techniques. We start by considering the FS transform in the NPRO model and by noticing that, as already claimed and proved in [YZ06], if the original 3-round public-coin HVZK proof is witness indistinguishable (WI)[5], then the transformed protocol is still WI, and of course the proof of WI is RO free.

Notice that as in [Lin15], $\mathcal{P}$ and $\mathcal{V}$ need a common hash function (modeled as an NPRO in the soundness proof) to run the protocol and this can be enforced through a setup (i.e., a non-programmable CRS [Pas03a], or a global hash function [CLP13]). The use of the FS transform in the NPRO model is not sufficient for our purposes. Indeed we want generality and the HVZK proof might not be witness indistinguishable. Moreover we should make a witness available to the simulator. We solve this problem by using the OR composition of 3-round perfect HVZK proofs proposed in [CDS94]. We will let the prover $\mathcal{P}$ for NIZK to prove that either $x \in L \vee \rho \in \Lambda$.

[4]As suggested by an anonymous reviewer of TCC 2016-A, to reduce the overhead of the Lindell's construction one could use the approach proposed in Groth's PhD thesis [Gro04], thus committing to a secret share of the challenge instead of the first round of the Σ-protocol.

[5]We use WI both to mean witness indistinguishable and witness indistinguishability.

We notice that in [CDS94] the proposed OR composition is proved to guarantee WI only when applied to two instances of the same language having a public-coin *perfect* HVZK proof. We can avoid this limitation using a generalization discussed already in [GMY06b, GMY06a] that allows the OR composition of different protocols for different languages relying on *computational HVZK* only.

7.1.2 Comparison

Here we compare the computational effort, both for the prover and the verifier, required to execute Lindell's NIZK argument, our NIZK argument and the FS one. The properties of the three transforms are summarized in Table 7.1. The cost for the prover can be found in Table 7.2, while the one for the verifier can be found in Table 7.3. The comparison of the computational effort is performed with respect to three Σ-protocols[6]. Roughly speaking, in the comparisons, we consider the CRS to contain an instance of the the language DH of Diffie-Hellman triples with respect to 1024-bit prime p_{CRS} and consider two Σ-protocols: the one to prove that a triples is Diffie-Hellman[7] with respect to a prime p, for which we consider the cases in which p is 1024-bit and 2048-bit long[8], and the Σ-protocol for graph isomorphism (GI). For the Σ-protocol for graph isomorphism, we count only the modular exponentiations and do not count other operations (e.g., random selection of a permutation and generation of the adjacency matrix of permuted graphs) since they are extremely efficient and clearly dominated by the cost of modular exponentiations. A detailed description of the Σ-protocols and of the way we measure the computational effort is found in Section 7.6.

The tables give evidence of the fact that while Lindell's transform on some specific cases can replace the FS transform by paying a small overhead, in other cases there is a significant loss in performance. Our transform instead remains very close to the FS transform both when considering the amount of computation and when considering the generality of the protocols that can be given as input to the transform.

Transform	$HVZK$ for Λ	$HVZK$ for L	Soundness	Model
Lindell [Lin14]	special + perfect	special + perfect	special	NPRO+CRS
This work	computational	computational	optimal	NPRO+CRS
FS	/	computational	classic	PRO

Table 7.1: Requirements for the proofs in input to the three transforms.

	DH		GI
Transform	$\lvert p\rvert = 1024$	$\lvert p\rvert = 2048$	n vertices
Lindell [Lin14]	$2 \mod p + 12 \mod p_{\text{CRS}}$	$2 \mod p + 20 \mod p_{\text{CRS}}$	$4n^2 \mod p_{\text{CRS}}$
This work	$2 \mod p + 4 \mod p_{\text{CRS}}$	$2 \mod p + 4 \mod p_{\text{CRS}}$	$4 \mod p_{\text{CRS}}$
FS	$2 \mod p$	$2 \mod p$	/

Table 7.2: Efficiency of the three transforms: modular exponentiations for the prover.

[6]We consider the same Σ-protocol discussed in [Lin15] and in addition we consider the one for Graph Isomorphism since it has the special property of having a very long first round that can be computed very efficiently.

[7]See Section 7.6 for a formal definition of the polynomial relation and the respective Σ-protocols.

[8]Clearly, in case p is such that $|p| < |p_{\text{CRS}}|$, then Lindell's transform has a slightly smaller number of exponentiations with respect to the number of exponentiations that we count in the tables.

	DH		GI
Transform	$\lvert p\rvert = 1024$	$\lvert p\rvert = 2048$	n vertices
Lindell [Lin14]	$4 \mod p + 12 \mod p_{\text{CRS}}$	$4 \mod p + 20 \mod p_{\text{CRS}}$	$4n^2 \mod p_{\text{CRS}}$
This work	$4 \mod p + 4 \mod p_{\text{CRS}}$	$4 \mod p + 4 \mod p_{\text{CRS}}$	$4 \mod p_{\text{CRS}}$
FS	$4 \mod p$	$4 \mod p$	/

Table 7.3: Efficiency of the three transforms: modular exponentiations for the verifier.

Which protocols can be given in input to the transform? We stress that our transform allows for additional proof systems to be used for instantiating the CRS and for obtaining a NIZK argument system. This is not only a theoretical progress. Indeed there exist efficient constructions such as the one of [Vis06] that is a variation of the one of [MP03]. The construction of [Vis06] is an efficient 3-round HVZK proof system with optimal soundness for a language L and is not a Σ-protocol for the corresponding relation Rel_L. For further details, see Sec. 7.7.

7.2 HVZK Proof Systems and Σ-Protocols

We will model a random oracle as a random function $\mathcal{O} : \{0,1\}^* \rightarrow \{0,1\}^n$. For simplicity, we will omit the modulus in modular arithmetic calculations.

Definition 28. *A 3-round proof or argument system $\Pi_L = (\mathcal{P}_L, \mathcal{V}_L)$ for $\mathcal{NP}$-language L is* Honest-Verifier Zero Knowledge (HVZK) *if there exists a PPT simulator algorithm* Sim *that takes as input security parameter 1^n and instance $x \in L$ and outputs an accepting transcript for x. Moreover, the distribution of the output of the simulator on input x is computationally indistinguishable from the distribution of the honest transcript obtained when $\mathcal{V}_L$ and $\mathcal{P}_L$ run Π_L on common input x and any private input w such that $(x, w) \in \mathsf{Rel}_L$.*

If the transcripts are identically distributed we say that Π_L is perfect HVZK.

Definition 29. *A 3-round public-coin proof system $\Pi_L = (\mathcal{P}_L, \mathcal{V}_L)$ for language L with challenge length l enjoys* optimal soundness *if for every $x \notin L$ and for every first-round message a there is at most one challenge $e \in \{0,1\}^l$ for which there exists a third-round message z such that (a, e, z) is accepting for x.*

Note that any 3-round public-coin optimally sound proof system with challenge length l has soundness error 2^{-l} [MP03].

Definition 30. *A 3-round public-coin protocol $\Pi_L = (\mathcal{P}_L, \mathcal{V}_L)$ with challenge length l is a Σ-protocol for an $\mathcal{NP}$-language L if it enjoys the following properties:*

- *Completeness.*
- *Special Soundness.*
- *Special Honest Verifier Zero Knowledge (special HVZK).*

It is easy to see that Σ-protocols enjoy optimal soundness. The converse, however, is not true. See Section 7.7 for an example of an optimal-sound 3-round public-coin proof system that does not enjoy special soundness (and is special perfect HVZK).

7.2.1 Challenge Lengths of 3-Round HVZK Proofs

Challenge-length amplification. The challenge of a 3-round public-coin proof system with HVZK and optimal soundness can be extended through parallel repetition.

Lemma 10. *Let Π_L be a 3-round public-coin proof system with optimal soundness for $\mathcal{NP}$-language L that enjoys perfect HVZK and has challenge length l. The protocol Π_L^k consisting of k parallel instances of Π_L is a 3-round public-coin proof system for relation L that enjoys perfect HVZK, has optimal soundness and has challenge length $k \cdot l$.*

Proof. The HVZK it is preserved by Π_L^k for the same arguments of [CDS94]. About the optimal soundness of Π_L^k, it is simple to see that if the protocol Π_L^k in not optimal sound then also Π_L is not optimal sound. □

A similar lemma can be proved for a Σ-protocol (as in [GMY06a, CPS+16a]) for which HVZK is not perfect.

Challenge-length reduction. We now show that starting from any 3-round public-coin proof system that enjoys HVZK and has optimal soundness with challenge length l, one can construct a 3-round public-coin proof system that still enjoys HVZK, has optimal soundness but works with a shorter challenge. Moreover perfect HVZK is preserved. A similar transformation was shown in [Dam10] for the case of Σ-protocol that are special perfect HVZK.

Lemma 11. *Let Π_L be a HVZK 3-round public-coin proof system for L with optimal soundness and challenge length l. Then for every $l' < l$, there exists a 3-round public-coin proof system Π'_L for L with HVZK and optimal soundness and challenge length l'. Protocol Π'_L has the same efficiency as Π_L and, moreover, if Π_L is perfect HVZK so is Π'_L.*

Proof. We now give a description of Π'_L.

Common input: instance x for an $\mathcal{NP}$-language L.

Private input of $\mathcal{P}'_L$: w s.t. $(x, w) \in \mathsf{Rel}_L$.

The protocol Π'_L:

1. $\mathcal{P}'_L$ computes $a \leftarrow \mathcal{P}_L(x, w)$ and sends it to $\mathcal{V}'_L$; [9]
2. $\mathcal{V}'_L$ randomly chooses challenge $e \leftarrow \{0,1\}^{l'}$ and sends it to $\mathcal{P}'_L$;
3. $\mathcal{P}'_L$ randomly chooses $pad \leftarrow \{0,1\}^{(l-l')}$, sets $e' = e|pad$, computes $z \leftarrow \mathcal{P}_L(x, w, a, e')$ and sends $z' = (z, pad)$ to $\mathcal{V}'_L$;
4. $\mathcal{V}'_L$ outputs the output of $\mathcal{V}_L(x, a, e|pad, z)$.

Completeness follows directly from the completeness of Π.

To prove the HVZK we can consider the simulator Sim', that on input x runs as follows:

1. run $(a, e', z) \leftarrow \mathsf{Sim}(x)$;
2. set pad equal to the last $l - l'$ bits of e', and set e equal to the fist l' bits of e';
3. output $(a, e, (z, pad))$.

[9]Because all the protocols of this chapter are public coin, we do not make a distinction between the verifier algorithm and the algorithm that decides whether to accept or not at the end of the interaction with the prover.

To conclude the proof we only observe that the optimal soundness follows directly from the optimal soundness of Π. $\square$

The following theorem follows from Lemma 10 and 11,

Theorem 22. *Suppose $\mathcal{NP}$-language L admits a HVZK 3-round public-coin proof system Π_L that has optimal soundness and challenge length l. Then for any $l' > 0$ there exists HVZK 3-round public-coin proof system Π'_L that has optimal soundness and challenge length l'. If $l' \leq l$ then Π'_L is as efficient as Π_L. Otherwise the communication and computation complexities of Π'_L are at most $\lceil l'/l \rceil$ times the ones of Π_L. Moreover, perfect HVZK is preserved.*

7.2.2 3-Round Public-Coin HVZK Proofs for OR Composition of Statements

In this section we recall the construction of [CDS94] that starts from a HVZK 3-round public-coin proof system Π_L for an $\mathcal{NP}$-language L and constructs a HVZK 3-round public-coin proof system $\Pi_{L\vee L}$ for the "OR" language of L; that is the $\mathcal{NP}$-language

$$L \vee L = \{(x_0, x_1) : x_0 \in L \vee x_1 \in L\}.$$

Below we give the descriptions of the prover $\mathcal{P}_{L\vee L}$ and of the verifier $\mathcal{V}_{L\vee L}$ of $\Pi_{L\vee L}$. In the description, we let Sim denote the simulator for Π_L and l denote the challenge length of Π_L. We also let $b \in \{0,1\}$ be such that w is a witness for $x_b \in L$; that is, $(x_b, w) \in \mathsf{Rel}_L$.

Common input: instances x_0, x_1 for an $\mathcal{NP}$-language L.

Private input of $\mathcal{P}_{L\vee L}$: w s.t $(x_0, x_1, w) \in \hat{\mathsf{Rel}}_{L\vee L}$. where

$$\hat{\mathsf{Rel}}_{L\vee L} = \left\{((x_0, x_1), w) : \left((x_0, w) \in \mathsf{Rel}_L \wedge x_1 \in \hat{L}\right) \vee \left((x_1, w) \in \mathsf{Rel}_L \wedge x_0 \in \hat{L}\right)\right\}.$$

The protocol $\Pi_{L\vee L}$:

1. $\mathcal{P}_{L\vee L}$ computes $a_b \leftarrow \mathcal{P}_L(x_b, w)$, $(a_{1-b}, e_{1-b}, z_{1-b}) \leftarrow \mathsf{Sim}(x_{1-b})$ and sends (a_0, a_1) to $\mathcal{V}_{L\vee L}$.
2. $\mathcal{V}_{L\vee L}$ chooses at random challenge $e \leftarrow \{0,1\}^l$ and sends e to $\mathcal{P}_{L\vee L}$.
3. $\mathcal{P}_{L\vee L}$ sets $e_b = e \oplus e_{1-b}$, computes $z_b \leftarrow \mathcal{P}_L(x_b, w, a_b, e_b)$ and outputs $\big((e_0, e_1), (z_0, z_1)\big)$.
4. $\mathcal{V}_{L\vee L}\big((x_0, x_1), (a_0, a_1), e, ((e_0, e_1), (z_0, z_1))\big)$. $\mathcal{V}_{L\vee L}$ accepts if and only if $e = e_0 \oplus e_1$ and $\mathcal{V}_L(x_0, a_0, e_0, z_0) = 1$ and $\mathcal{V}_L(x_1, a_1, e_1, z_1) = 1$.

Theorem 23 ([CDS94, GMY06b])**.** *If Π_L is a HVZK 3-round public-coin proof system with optimal soundness for $\mathcal{NP}$-language L then $\Pi_{L\vee L}$ is a HVZK 3-round public-coin proof system with optimal soundness for $\mathcal{NP}$-language $L \vee L$ and is WI for polynomial-time relation*

$$\mathsf{Rel}_{L\vee L} = \left\{((x_0, x_1), w) : \left((x_0, w) \in \mathsf{Rel}_L \wedge x_1 \in L\right) \vee \left((x_1, w) \in \mathsf{Rel}_L \wedge x_0 \in L\right)\right\}.$$

Moreover if Π_L is perfect HVZK then $\Pi_{L\vee L}$ is perfect WI for polynomial-time relation $\hat{\mathsf{Rel}}_{L\vee L}$

We remark that results of [CDS94, GMY06b] are known to hold for Σ-protocols, but in the proof of WI they use only HVZK. Therefore their results also hold starting from a HVZK 3-round public-coin proof system with optimal soundness (and not necessarily special soundness) that we consider in the above theorem. Indeed we observe that $\Pi_{L\vee L}$ has optimal soundness for the following reason. Suppose that $\Pi_{L\vee L}$ does not enjoy optimal soundness. This means that for a false instance and the same first round (a_0, a_1) there are two accepting conversation, namely:

$$\Big((a_0, a_1), e, ((e_0, e_1), (z_0, z_1))\Big), \Big((a_0, a_1), e', ((e'_0, e'_1), (z'_0, z'_1))\Big)$$

with $e \neq e'$. Then it must be the case that for some $b = 0$ or $b = 1$, $e_b \neq e'_b$ and then (a_b, e_b, z_b) (a_b, e'_b, z'_b) are two accepting transcripts with the same first round for the protocol Π_L, and thus the optimal soundness of Π_L is violated.

It is possible to extend the above construction to handle two different $\mathcal{NP}$-languages L_0, L_1 that admit HVZK 3-round public-coin proof system with optimal soundness. Indeed by Theorem 22, we can assume, without loss of generality, that L_0 and L_1 have 3-round public-coin proof systems Π_{L_0} and Π_{L_1} with the same challenge length.

Assuming that L_0 and L_1 have 3-round public-coin proof systems Π_{L_0} and Π_{L_1} that are HVZK and have optimal soundness with the same challenge length. We can apply the same construction outlined above to obtain a 3-round public-coin proof system $\Pi_{L_0\vee L_1}$ that enjoys HVZK and has optimal soundness for relation

$\hat{\mathsf{Rel}}_{L_0\vee L_1} = \Big\{\big((x_0, x_1), w\big) : \big((x_0, w) \in \mathsf{Rel}_{L_0} \wedge x_1 \in \hat{L}_1\big) \vee \big((x_1, w) \in \mathsf{Rel}_{L_1} \wedge x_0 \in \hat{L}_0\big)\Big\}$.

We have the following theorem.

Theorem 24. *If Π_{L_0} and Π_{L_1} are HVZK 3-round public-coin proof systems with optimal soundness for $\mathcal{NP}$-languages L_0 and L_1 then $\Pi_{L_0\vee L_1}$ is a HVZK 3-round public-coin proof system with optimal soundness for the for $\mathcal{NP}$-language*

$$L_0 \vee L_1 = \{(x_0, x_1) : x_0 \in L_0 \vee x_1 \in L_1\}$$

and is WI for polynomial-time relation

$$\mathsf{Rel}_{L_0\vee L_1} = \Big\{((x_0, x_1), w) : \big((x_0, w) \in \mathsf{Rel}_{L_0} \wedge x_1 \in L_1\big) \vee \big((x_1, w) \in \mathsf{Rel}_{L_1} \wedge x_0 \in L_0\big)\Big\}.$$

Moreover, if Π_{L_0} and Π_{L_1} are perfect then $\Pi_{L_0\vee L_1}$ is perfect WI for polynomial-time relation $\hat{\mathsf{Rel}}_{L\vee L}$.

7.3 Non-Interactive Argument Systems

Some definitions presented in this section are taken from [Lin15].

Definition 31. *A* non-interactive argument system *for an $\mathcal{NP}$-language L consists of three PPT machines $(\mathcal{CRS}, \mathcal{P}, \mathcal{V})$, that have the following properties:*

- *Completeness: for all $(x, w) \in \mathsf{Rel}_L$, it holds that:*

$$\text{Prob}\left[\, \sigma \leftarrow \mathcal{CRS}(1^n); \mathcal{V}(\sigma, x, \mathcal{P}(\sigma, x, w)) = 1 \,\right] = 1.$$

- *Adaptive Soundness: for every PPT function $f : \{0,1\}^{poly(n)} \rightarrow \{0,1\}^n \setminus L$ for all PPT prover $\mathcal{P}^\star$, there exists a negligible function ν, such that for all n:*

$$\text{Prob}\left[\, \sigma \leftarrow \mathcal{CRS}(1^n); \mathcal{V}^{\mathcal{O}}(\sigma, f(\sigma), \mathcal{P}^{\star\mathcal{O}}(\sigma)) = 1 \,\right] \leq \nu(n)$$

where $\mathcal{O} : \{0,1\}^* \rightarrow \{0,1\}^n$ is a random function.

Definition 32. *A non-interactive argument system is* adaptive unbounded zero knowledge *(NIZK) for an $\mathcal{NP}$-language L if there exists a probabilistic PPT simulator S such that for every PPT function*

$$f : \{0,1\}^{\texttt{poly(n)}} \to \Big(\{0,1\}^n \times \{0,1\}^{\texttt{poly(n)}}\Big) \cap \mathsf{Rel}_L,$$

and for every PPT malicious verifier $\mathcal{V}^\star$, there exists a negligible function ν such that,

$$\Big|\mathrm{Prob}\Big[\ \mathcal{V}^\star\Big(R_f(\mathcal{P}^f(n,p))\Big) = 1\ \Big] - \mathrm{Prob}\left[\ \mathcal{V}^\star\left(S_f(n,p)\right) = 1\ \right]\Big| \le \nu(n)$$

where f_1 and f_2 denote the first and second output of f, respectively, and $R_f(\mathcal{P}^f(n,p))$ and $S_f(n,p)$ denote the output from the following experiments:

Real proofs *$R_f(\mathcal{P}^f(n,p))$:*

- *$\sigma \leftarrow \mathcal{CRS}(1^n)$ a common reference string is sampled.*
- *For $i = 1, \dots, p(n)$ (initially $\vec{x}$ and $\vec{\pi}$ are empty):*
  - *$x_i \leftarrow f_1(\sigma, \vec{x}, \vec{\pi})$: the next statement x_i to be proven is chosen.*
  - *$\pi_i \leftarrow \mathcal{P}(\sigma, f_1(\sigma,\vec{x},\vec{\pi}), f_2(\sigma,\vec{x},\vec{\pi}))$: the ith proof is generated.*
  - *set $\vec{x} = x_1 \dots x_i$ and $\vec{\pi} = \pi_1 \dots \pi_i$.*
- *output $(\sigma, \vec{x}, \vec{\pi})$.*

Simulation *$S_f(n,p)$:*

- *$\sigma \leftarrow S(1^n)$ a common reference string is sampled.*
- *For $i = 1, \dots, p(n)$ (initially $\vec{x}$ and $\vec{\pi}$ are empty):*
  - *$x_i \leftarrow f_1(\sigma, \vec{x}, \vec{\pi})$: the next statement x_i to be proven is chosen.*
  - *$\pi_i \leftarrow S(x_i)$: simulator S generates a simulated proof π_i that $x_i \in L$.*
  - *set $\vec{x} = x_1 \dots x_i$ and $\vec{\pi} = \pi_1 \dots \pi_i$.*
- *output $(\sigma, \vec{x}, \vec{\pi})$.*

Definition 33. *A non-interactive argument system is* adaptive unbounded witness indistinguishable *(NIWI) for an $\mathcal{NP}$-language L if for every PPT adversary $\mathcal{V}^\star$, for every PPT function*

$$f : \{0,1\}^{\texttt{poly(n)}} \to \Big(\{0,1\}^n \times \{0,1\}^{\texttt{poly(n)}} \times \{0,1\}^{\texttt{poly(n)}}\Big) \cap \mathsf{Rel}_L^{\wedge},$$

and for every polynomial $p(\cdot)$, there exists a negligible function ν such that

$$\Big|\mathrm{Prob}\Big[\ \mathcal{V}^\star(R_0^{\mathcal{P},f}(n,p)) = 1\ \Big] - \mathrm{Prob}\Big[\ \mathcal{V}^\star(R_1^{\mathcal{P},f}(n,p)) = 1\ \Big]\Big| \le \nu(n),$$

where $\mathsf{Rel}_L^{\wedge} = \{(x, w^0, w^1) : (x, w^0) \in \mathsf{Rel}_L \wedge (x, w^1) \in \mathsf{Rel}_L\}$ and $R_b^{\mathcal{P},f}$ is the following experiment. $R_b^{\mathcal{P},f}(n,p)$:

- *$\sigma \leftarrow \mathcal{CRS}(1^n)$.*
- *For $i = 1, \dots, p(n)$ (initially $\vec{x}$ and $\vec{\pi}$ are empty):*
  - *$(x_i, w_i^0, w_i^1) \leftarrow f(\sigma, \vec{x}, \vec{\pi})$:*
    statement x_i to be proven and witnesses w_i^0, w_i^1 for x_i are generated.
  - *$\pi_i \leftarrow \mathcal{P}(\sigma, x_i, w_i^b)$: the ith proof is generated.*
  - *set $\vec{x} = x_1 \dots x_i$ and $\vec{\pi} = \pi_1 \dots \pi_i$.*
- *output $(\sigma, \vec{x}, \vec{\pi})$.*

7.4 NIWI Argument Systems from 3-Round HVZK Proofs

In this section we discuss the FS transform in the NPRO model in order to obtain a NIWI argument system $\Pi = (\mathcal{P}, \mathcal{V})$ for a polynomial relation Rel_L. We start from a 3-round public-coin WI HVZK proof system with optimal soundness $\Pi_L = (\mathcal{P}_L, \mathcal{V}_L)$ for the $\mathcal{NP}$ language L. $\mathcal{P}$ and $\mathcal{V}$ have access to an NPRO $H : \{0,1\}^* \rightarrow \{0,1\}^n$. We describe Π below and we assume that the challenge length of Π_L is the security parameter n.

Common input: instance x for $\mathcal{NP}$-language L.

Private input to $\mathcal{P}$: w s.t. $(x, w) \in \mathsf{Rel}_L$.

Common reference string: $\mathcal{CRS}$ samples a key s for a hash function family H and sets $\sigma = s$.

1. $\mathcal{P} \rightarrow \mathcal{V}$: The prover $\mathcal{P}$ executes the following steps:
   1.1. $a \leftarrow \mathcal{P}_L(x, w)$;
   1.2. $e \leftarrow H_s(x, a)$;
   1.3. $z \leftarrow \mathcal{P}_L(x, w, a, e)$;
   1.4. send $\pi = (a, e, z)$ to $\mathcal{V}$.
2. $\mathcal{V}$'s output: $\mathcal{V}$ outputs 1 if and only if $\mathcal{V}_L(x, a, e, z) = 1$ and $e = H_s(x, a)$.

The following theorem was proved by Yung and Zhao in [YZ06] (see Claim 1, page 4). For sake of completeness, we provide a proof of the claim below.

Theorem 25 ([YZ06]). *Let Π_L be a 3-round public-coin WI proof system for the polynomial relation Rel_L. Then Π is adaptive WI for Rel_L in the CRS model.*

Proof. We show that Π is adaptive WI for Rel_L through the following hybrids.

1. $\mathcal{H}_1$ is the experiment $R_0^{\mathcal{P},f}(n, p)$ (Definition 33), where $\mathcal{P}$ for $j = 1, \dots, p(n)$ executes Π and outputs π_j using the first of the two witnesses given in output by f.
2. $\mathcal{H}_i$ (with $i > 0$) differs from $\mathcal{H}_1$ in the first i interactions, where $\mathcal{P}$ executes Π using the second witness given in output by f. Namely: $\mathcal{P}$ on input (x_j, w_j^1) executes Π and outputs π_j using w_j^1 for all $j : 1 \leq j < i$. Instead, for the interactions $i \leq j < p(n) + 1$, $\mathcal{P}$ on input (x_j, w_j^0) executes Π using w_j^0 as a witness and outputs π_j.
3. $\mathcal{H}_{p(n)+1}$ is the experiment $R_1^{\mathcal{P},f}(n, p)$ (Definition 33), where $\mathcal{P}$ for $j = 1, \dots, p(n)$ executes Π and outputs π_j using the second witness given in output by f.

$\mathcal{H}_i \approx \mathcal{H}_{i+1}$: Suppose there exists a malicious adversary $\mathcal{V}^\star$ that distinguishes between the experiments $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ with $1 \leq i \leq p(n)$, then we can show that there exists an adversary $\mathcal{A}$ that breaks the WI property of Π_L. The reduction works as follows.

1. For $j = 1, \dots, i-1$, $\mathcal{A}$ on input (x_j, w_j^1) executes Π using w_j^1 to obtain π_j.
2. For $j = i$, $\mathcal{A}$ interacts with the WI challenger of Π_L as follows:
   2.1. $\mathcal{A}$ has on input (x_j, w_j^0, w_j^1) and sends it to the challenger of WI;
   2.2. the challenger computes and sends the first message a_j to $\mathcal{A}$;

2.3. $\mathcal{A}$ computes $e_j = H_s(a_j)$ and sends it to the challenger of WI;

2.4. the challenger computes and sends z_j to $\mathcal{A}$;

2.5. $\mathcal{A}$ sends $\pi_j = (a_j, e_j, z_j)$ to $\mathcal{V}^\star$;

2.6. $\mathcal{A}$ adds to $\vec{x}$ the theorem x_j and to $\vec{\pi}$ the proof π_j.

3. $\forall j = i+1, \ldots, p(n)$ $\mathcal{A}$ on input (x_j, w_j^0) executes Π using w_j^0 to obtain π_j.

4. Set $\vec{x} = x_1, \ldots, x_{p(n)}$ and $\vec{\pi} = \pi_1, \ldots, \pi_{p(n)}$.

$\mathcal{A}$ sends $\vec{x}$ and $\vec{\pi}$ to $\mathcal{V}^\star$ and outputs what $\mathcal{V}^\star$ outputs.

We now observe that if the challenger of WI has used the first witness we are in $\mathcal{H}_i$ otherwise we are in $\mathcal{H}_{i+i}$. The security proof ends with the observation that $R_0^{\mathcal{P},f}(n,p) \equiv \mathcal{H}_1 \approx \cdots \approx \mathcal{H}_{p(n)} \approx \mathcal{H}_{p(n)+1} \equiv R_1^{\mathcal{P},f}(n,p)$. □

Adaptive soundness. To prove soundness, we follow [Lin15] and use the fact that, for every function g, with a sufficiently large co-domain, relation $\mathsf{Rel} = \{(x, g(x))\}$ is evasive [CGH04] in the NPRO model. A relation Rel is *evasive* if, given access to a random oracle $\mathcal{O}$, it is infeasible to find a string x so that the pair $(x, \mathcal{O}(x)) \in \mathsf{Rel}$.

Theorem 26. *Let Π_L be a 3-round public-coin proof system with optimal soundness for the $\mathcal{NP}$-language L, and let H be a non programmable random oracle. Then, Π is a non-interactive argument system with (adaptive) soundness for L in the NPRO model.*

Proof. Completeness of Π follows from the completeness of Π_L. Let $\mathcal{O}$ be an NPRO. In order to prove the soundness of Π we use the fact that for any function g, the relation $\mathsf{Rel} = \{(x, g(x))\}$ is evasive. We define the function g s.t. $g(x, a) = e$, where there exists z such that the transcript (a, e, z) is accepting for the instance x. If $x \notin L$ by the optimal soundness property we have that for every a there is a single e for which there is some z so that (a, e, z) is accepting. Therefore g is a function, as required and it follows that the relation $\mathsf{Rel} = \{((x, a), g(x, a))\}$ is evasive.

Suppose that there exists a polynomial function f and a malicious prover $\mathcal{P}^\star$ such that $\mathcal{P}^\star$ proves a false statement (i.e., $\mathcal{V}^{\mathcal{O}}(\sigma, f(\sigma), \mathcal{P}^{\star\mathcal{O}}(\sigma)) = 1$, where $\sigma \leftarrow \mathcal{CRS}(1^n)$) with non-negligible probability, then there is an adversary $\mathcal{A}$ that finds (x, a) s.t. $\mathcal{O}(x, a) = g(x, a)$ with non-negligible probability. The adversary $\mathcal{A}$ works as follows. First, it runs $\sigma \leftarrow \mathcal{CRS}(1^n)$. Then it runs $(x, a, e, z) \leftarrow \mathcal{P}^\star(\sigma)$. Finally it outputs $(x, \mathcal{O}(x, a))$. From the contradicting assumption we know that $\mathcal{V}^{\mathcal{O}}(\sigma, f(\sigma), (a, e, z)) = 1$ with non-negligible probability. This implies that the transcript $(a, \mathcal{O}(x, a), z)$ is accepting with non-negligible probability. Since $x \notin L$ there exists only one e for which $(a, \mathcal{O}(x, a), z)$ is accepting. Therefore we have that with non-negligible probability it holds that $\mathcal{O}(x, a) = e$ (i.e., $\mathcal{O}(x, a) = g(x, a)$) and this contradicts the fact that any function g is evasive for an NPRO. □

7.5 Our Transform: Non-Interactive Zero Knowledge from HVZK

From the previous section we know that if we have a 3-round HVZK proof system with optimal soundness $\Pi_{L\vee\Lambda} = (\mathcal{P}_{L\vee\Lambda}, \mathcal{V}_{L\vee\Lambda})$ for polynomial relation

$$\hat{\mathsf{Rel}}_{L\vee\Lambda} = \{((x, \rho), w) : ((x, w) \in \mathsf{Rel}_L \wedge \rho \in \hat{\Lambda}) \vee ((\rho, \omega) \in \mathsf{Rel}_\Lambda \wedge x \in \hat{L})\}$$

that is also WI for polynomial relation

$$\mathsf{Rel}_{L\vee\Lambda} = \{((x, \rho), w) : ((x, w) \in \mathsf{Rel}_L \wedge \rho \in \Lambda) \vee ((\rho, w) \in \mathsf{Rel}_\Lambda \wedge x \in L)\},$$

then we can apply the FS transform to make it non-interactive while preserving WI and soundness[10].

Here we make use of the above result in order to transform a 3-round HVZK proof system with optimal soundness for an $\mathcal{NP}$-language L into a NIZK argument for L in the CRS model using an NPRO in the proof of soundness. First, we recall the notion of Membership-hard languages with efficient sampling that will be used in our final construction.

Membership-hard languages with efficient sampling. Lindell defines a membership-hard language Λ as a language such that one can efficiently sample both instances that belong to the language and instances that do not belong to the language. Still distinguishing among these two types of instances is hard. This is formalized through a sampling algorithm S_Λ that on input a bit b outputs an instance $\rho \in \Lambda$ along with a witness ω when $b = 0$, and outputs an instance $\rho \notin \Lambda$ otherwise. No polynomial-time distinguisher on input ρ can guess b with probability non-negligibly better than $1/2$. Let S^ρ_Λ denote the instance part of the output (i.e., without the witness when b is 0).

Definition 34 ([Lin15])**.** *Let Λ be a language. We say that Λ is membership-hard with efficient sampling if there exists a PPT sampler S_Λ such that for every PPT distinguisher $\mathcal{D}$ there exists a negligible function μ such that:* $|\text{Prob}\left[\, \mathcal{D}(S^\rho_\Lambda(0, 1^n), 1^n) = 1 \,\right] - \text{Prob}\left[\, \mathcal{D}(S_\Lambda(1, 1^n), 1^n) = 1 \,\right]| \leq \mu(n)$.

Now we are ready to show our NIZK argument $\Pi = (\mathcal{P}, \mathcal{V})$.

Common input: instance x for an $\mathcal{NP}$-language L.

Private input of $\mathcal{P}$: w s.t $(x, w) \in \mathsf{Rel}_L$.

Common reference string: $\mathcal{CRS}$ on input 1^n runs $\rho \leftarrow S_\Lambda(1, 1^n)$ where Λ is an membership-hard language and samples a key s for a hash function family H. Then it sets $\sigma = (\rho, s)$.

$\mathcal{P} \rightarrow \mathcal{V}$: $\mathcal{P}$ executes the following steps:

1. $a \leftarrow \mathcal{P}_{L\vee\Lambda}((x, \rho), w)$;
2. $e \leftarrow H_s(x, a)$;
3. $z \leftarrow \mathcal{P}_{L\vee\Lambda}((x, \rho), w, a, e)$;
4. send $\pi = (a, e, z)$ to $\mathcal{V}$.

$\mathcal{V}'s$ output: $\mathcal{V}$ accepts if and only if $\mathcal{V}_{L\vee\Lambda}((x, \rho), a, e, z) = 1$ and $e = H_s(x, a)$.

In our construction we suppose that the challenge length of Π_Λ is n, where n denotes the security parameter. Therefore to use the OR composition of [CDS94] we need to consider a 3-round public-coin proof system with HVZK and optimal soundness Π_L for Rel_L that has challenge length n (and therefore soundness error 2^{-n}). This is not a problem because we can use Theorem 22 to transform every 3-round public-coin proof system with HVZK and optimal soundness with challenge n' (where $n' \neq n$) to another one with challenge length n. More precisely, if $n' > n$ we can use Lemma 11 to reduce n' to n almost for free. If $n' < n$ we need to use Lemma 10, therefore we have to run multiple executions of Π_L to apply the OR composition of [CDS94]. Notice that this potential computational effort is implicit also for the FS transform and for Lindell's transform. Indeed if the original 3-round public-coin proof system with HVZK and optimal soundness has just a one-bit (or in general a short) challenge then clearly the

[10]We recall that the common hash function is modelled as a NPRO only in the proof of soundness.

resulting NIZK is not sound. Therefore the parallel repetition of the 3-round public-coin proof system with HVZK and optimal soundness is required before applying the transform in order to reduce the soundness error (see Section 7.2.1).

Theorem 27. *Let $\Pi_{L\vee\Lambda}$ be a 3-round public-coin proof system for polynomial relation $\hat{\mathsf{Rel}}_{L\vee\Lambda}$ that is WI for polynomial relation $\mathsf{Rel}_{L\vee\Lambda}$. Then Π is zero knowledge for Rel_L in the CRS model.*

Proof. The simulator S works as follows:

1. S on input 1^n, runs $(\rho, \omega) \leftarrow S_\Lambda(0, 1^n)$; samples a key s for a hash function and sets $\sigma = \{\rho, s\}$ and outputs σ.

2. S on input σ, ω and x_i (for every $i = 1, \dots, p(n)$) computes $a \leftarrow \mathcal{P}_{L\vee\Lambda}((x_i, \rho), \omega)$, $e \leftarrow H_s(x_i, a)$ and $z \leftarrow \mathcal{P}_{L\vee\Lambda}((x_i, \rho), \omega, a, e)$. It outputs $\pi_i = (a, e, z)$.

We show that the output of S is computationally indistinguishable from a real transcript given in output by $\mathcal{P}$ in a real execution of Π through the following hybrids games.

1. $\mathcal{H}_0$ is the experiment $R_f(\mathcal{P}^f(n, p))$ (Definition 32).

2. $\mathcal{H}_1$ differs from $\mathcal{H}_0$ in the way that ρ is generated. Indeed in $\mathcal{H}_1$ we have that σ is computed by running $S_\Lambda(0, 1^n)$. The second output ω of S_Λ is not used. Clearly $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable otherwise the membership-hard property of Λ would be contradicted. More details on this reduction will be given below.

3. $\mathcal{H}_2$ differs from $\mathcal{H}_1$ just on the witness used by $\mathcal{P}_{L\vee\Lambda}$. Indeed now ω is used as witness. The WI property of $\Pi_{L\vee\Lambda}$ guarantees that $\mathcal{H}_2$ can not be distinguished from $\mathcal{H}_1$. More details on this reduction will be given below. Notice that $\mathcal{H}_2$ corresponds to the simulation.

$\mathcal{H}_0 \approx \mathcal{H}_1$: If there exists a malicious verifier $\mathcal{V}^\star$ that distinguishes between $\mathcal{H}_0$ and $\mathcal{H}_1$, then there exists an adversary $\mathcal{A}$ that breaks the membership-hard property of Λ. The reduction works as follows.

1. $\mathcal{A}$ queries the challenger of S_Λ that sends back ρ.

2. $\mathcal{A}$ samples a key s for a hash function family H and sets $\sigma = \{\rho, s\}$.

3. $\mathcal{A}$ on input $(x_i, w_i) \in \mathsf{Rel}_L$ for $i = 1, \dots, p(n)$ computes the following steps:
   3.1. compute $a_i \leftarrow \mathcal{P}_{L\vee\Lambda}((x_i, \rho), w_i)$;
   3.2. compute $e_i \leftarrow H_s(x_i, a_i)$;
   3.3. compute $z_i \leftarrow \mathcal{P}_{L\vee\Lambda}((x_i, \rho), w_i, a_i, e_i)$;
   3.4. set $\pi_i = (a_i, e_i, z_i)$;
   3.5. set $\vec{x} = x_1, \dots, x_i$ and $\vec{\pi} = \pi_1, \dots, \pi_i$.

4. $\mathcal{A}$ sends $\sigma, \vec{x}, \vec{\pi}$ to $\mathcal{V}^\star$.

5. $\mathcal{A}$ outputs the output of $\mathcal{V}^\star$.

We now observe that if the challenger of a sampling algorithm S_Λ sends $\rho \notin \Lambda$ we are in $\mathcal{H}_0$ otherwise we are in $\mathcal{H}_1$. This implies that $\mathcal{H}_0 \approx \mathcal{H}_1$.

$\mathcal{H}_1 \approx \mathcal{H}_2$**:** If there exists a distinguisher $\mathcal{V}^\star$ that distinguishes between $\mathcal{H}_1$ and $\mathcal{H}_2$, then there exists an adversary $\mathcal{A}$ against the adaptive NIWI property of $\Pi_{L\vee\Lambda}$, therefore contradicting Theorem 25. The reduction works as follows.

1. $\mathcal{A}$ runs $(\rho, \omega) \leftarrow S_\Lambda(0, 1^n)$, samples a key s for a hash function and sets $\sigma = \{\rho, s\}$.
2. $\mathcal{A}$ has on input a PPT function $f = (f_1, f_2)$ and defines $f' = (f_1', f_2')$ as follows: $f'(\sigma, \vec{t}, \vec{\pi})$ on input a CRS σ, a vector of theorems $\vec{t} = (x_1, \rho), \ldots, (x_{p(n)}, \rho)$ and a vector of proofs $\vec{\pi} = \pi_1, \ldots, \pi_{p(n)}$ returns $(f_1(\sigma, \vec{x}, \vec{\pi}), \rho), (f_2(\sigma, \vec{x}, \vec{\pi}), \omega)$.
3. $\mathcal{A}$ interacts with the challenger of adaptive NIWI, using f', in order to obtain x_i, $\pi_i = \{a_i, e_i, z_i\}$, for $i = 1, \ldots, p(n)$.
4. $\mathcal{A}$ sets $\vec{x} = x_1, \ldots, x_{p(n)}$ and $\vec{\pi} = \pi_1, \ldots, \pi_{p(n)}$.
5. $\mathcal{A}$ sends $\sigma, \vec{x}, \vec{\pi}$ to $\mathcal{V}^\star$ and outputs the output of $\mathcal{V}^\star$.

We now observe that if the challenger of NIWI uses the first witness w_i we are in $\mathcal{H}_1$ otherwise we are in $\mathcal{H}_2$. This implies that $\mathcal{H}_1 \approx \mathcal{H}_2$.

We can thus conclude that $\mathcal{H}_0 \approx \mathcal{H}_1 \approx \mathcal{H}_2$ and therefore the output of S is computational indistinguishable from a real transcript. □

Theorem 28. *Let $\Pi_{L\vee\Lambda}$ be a 3-round public-coin HVZK proof system with optimal soundness for relation $\mathsf{Rel}_{L\vee\Lambda}$, and WI for relation $\hat{\mathsf{Rel}}_{L\vee\Lambda}$, and let H be an NPRO. Then, Π is a non-interactive argument system with adaptive soundness for the relation Rel_L in the CRS model using the NPRO model for soundness.*

Proof. The completeness of Π follows from the completeness of $\Pi_{L\vee\Lambda}$. In order to prove adaptive soundness we notice that an adversarial prover proving a false statement $x \notin L$ can be directly reduced to an adversarial prover proving a false statement for $\Pi_{L\vee\Lambda}$ in the NPRO model. This contradicts Theorem 26. Indeed the only subtlety that is worthy to note is that when the adversarial prover runs the protocol, we have that the statement "$\rho \in \Lambda$" stored in the CRS is false, therefore if also the instance "$x \notin L$" proved by the prover is false then the OR composition of the two statements is also false. □

7.6 Efficiency Comparison

In this section we illustrate in details Tables, 7.2 and 7.3 of Section 7.1.2 has been counted. First of all we need to briefly introduce two Σ-protocols, one to prove that a tuple is DH($\Pi_{\mathcal{DH}}$ [HL10]), and the other one to prove that two graphs are isomorphic ($\Pi_{\mathcal{GH}}$ [GMW86]). Our comparison assumes that the CRS is a DH tuple $((G_{\text{CRS}}, q_{\text{CRS}}, p_{\text{CRS}}, g_{\text{CRS}}), A_{\text{CRS}}, B_{\text{CRS}}, C_{\text{CRS}})$ with p_{CRS} and q_{CRS} primes such that $p_{\text{CRS}} = 2q_{\text{CRS}} + 1$ and $|p_{\text{CRS}}| = 1024$. We distinguish two cases. In the first one the prover wants to prove that a tuple $((G, q, p, g), A, B, C)$ is a DH tuple, and in the other one the prover tries to convince the verifier that two graphs G_0 and G_1 with n vertices each are isomorphic.

A Σ-protocol for Diffie-Hellman tuples. We consider the following polynomial-time relation

$$\mathsf{Rel}_{\mathcal{DH}} = \Big\{ \big(((G, q, g), A = g^r, B = h, C = h^r), r\big) : B^r = C \Big\}$$

over cyclic groups G of prime-order q. Typically, G is the subgroup of quadratic residues of $\mathbb{Z}_p$ for prime $p = 2q + 1$. We next briefly describe Σ-protocol $\Pi_{\mathcal{DH}} = (\mathcal{P}_{\mathcal{DH}}, \mathcal{V}_{\mathcal{DH}})$ for $\mathsf{Rel}_{\mathcal{DH}}$.

Common input: instance x and language DH.

Private input of $\mathcal{P}_{\mathcal{DH}}$: r.

The protocol $\Pi_{\mathcal{DH}}$:

1. $\mathcal{P}_{\mathcal{DH}}$ picks $t \in \mathbb{Z}_q$ at random, computes and sends $a = g^t$, $b = h^t$ to $\mathcal{V}_{\mathcal{DH}}$;

2. $\mathcal{V}_{\mathcal{DH}}$ chooses a random challenge $e \in \mathbb{Z}_q$ and sends it to $\mathcal{P}_{\mathcal{DH}}$;

3. $\mathcal{P}_{\mathcal{DH}}$ computes and sends $z = t + er$ to $\mathcal{V}_{\mathcal{DH}}$;

4. $\mathcal{V}_{\mathcal{DH}}$ accepts iff:
$$g^z = a \cdot A^e \texttt{ AND } h^z = b \cdot C^e.$$

We show the special HVZK simulator Sim for $\Pi_{\mathcal{DH}}$. Sim, on input x and a challenge e of length $|q| - 1$ executes the following steps:

1. randomly chooses $z \in \mathbb{Z}_q$;

2. computes $a = g^z \cdot A^{-e}$;

3. computes $b = h^z \cdot C^{-e}$.

Graph isomorphism. We show a Σ-protocol $\Pi_{\mathcal{GH}} = (\mathcal{P}_{\mathcal{GH}}, \mathcal{V}_{\mathcal{GH}})$ to prove that two graphs are isomorphic. Given two graphs G_0 and G_1, prover $\mathcal{P}_{\mathcal{GH}}$ wants to convince verifier $\mathcal{V}_{\mathcal{GH}}$ that he knows a permutation ϕ such that $\phi(G_0) = G_1$.

Common input: theorem $x = (G_0, G_1)$.

Private input of $\mathcal{P}_{\mathcal{GH}}$: ϕ.

The protocol $\Pi_{\mathcal{GH}}$:

1. $\mathcal{P}_{\mathcal{GH}}$ randomly chooses a permutation ψ and a bit $b \in \{0, 1\}$, computes and sends $P = \psi(G_b)$;

2. $\mathcal{V}_{\mathcal{GH}}$ chooses and sends a random bit $b' \in \{0, 1\}$ $\mathcal{P}_{\mathcal{GH}}$;

3. $\mathcal{P}_{\mathcal{GH}}$ sends the permutation τ to $\mathcal{V}_{\mathcal{GH}}$, where
$$\tau = \begin{cases} \psi & if\ b = b' \\ \psi\phi^{-1} & if\ b = 0, b' = 1 \\ \psi\phi & if\ b = 1, b' = 0 \end{cases}$$

4. $\mathcal{V}_{\mathcal{GH}}$ accepts if and only if $P = \tau(G_{b'})$.

Computational effort: two cases. We show a summary of the comparison among our transform and Lindell's transform in Tables 7.2 and 7.3. The cost is measured by considering the computations in terms of number of exponentiations made by $\mathcal{P}$ and of $\mathcal{V}$. In our comparison we consider that a CRS contains a DH tuple $((G_{\text{CRS}}, q_{\text{CRS}}, p_{\text{CRS}}, g_{\text{CRS}}), A_{\text{CRS}}, B_{\text{CRS}}, C_{\text{CRS}})$ with $|p_{\text{CRS}}| = n = 1024$, with security parameter n (therefore $|q_{\text{CRS}}| = 1023$). We consider two cases. In the first one we use the NIZK argument to prove that a tuple $((G, q, p, g), A, B, C)$ is a DH tuple; in particular we take in account two sub-cases: when $p = 1024$ and when $p = 2048$. In the second case we use the NIZK argument to prove the isomorphism between two graphs G_0 and G_1, and we assume that $k = n^2$ bits are needed to represent a graph with n vertices. We stress that Lindell's transform needs to commit the first round of a Σ-protocol (plus the instance to be proved, but for our comparison we ignore that the instance has to be committed) associated to the language that we take into account (the language of the DH tuples or the language of the isomorphic graphs). Therefore, using the described CRS, to commit to a string of 1023 bit, 4 exponentiations are required. This is a consequence of the fact that the commitment is made by executing the simulator associated with $\Pi_{\mathcal{DH}}$ (with $|q_{\text{CRS}}| = 1023$).

Case 1: proving that a tuple is a DH tuple.

- [Lin15]. When the instance to be proved is $((G, q, p, g), A, B, C)$ with $p = 1024$, the prover $\mathcal{P}$ needs to compute $a = g^t$, $b = h^t$ (as describe before) and needs to commit to them. The total size of a and b is 2048 bits, therefore to commit to 2048 bits we need to execute the DM commitment 3 times. This implies that the prover needs to compute $3 \cdot 4$ exponentiations mod p_{CRS} and 2 exponentiations mod p. The verifier $\mathcal{V}$ needs to checks if open of the DM commitments was correct, and also needs to compute $g^z = a \cdot A^e p$ and $h^z = b \cdot C^e$. For this reason the verifier needs to compute $3 \cdot 4$ exponentiations mod p_{CRS} plus 4 exponentiations mod p. With the same arguments we can count the amount of exponentiations needed to prove that the instance is a DH tuple with $p = 2048$.

- Our transform. When $|p| = 1024$ (resp., $|p| = 2048$) the prover need to run the simulator Sim of $\Pi_{\mathcal{DH}}$ with the instance $((G_{\text{CRS}}, q_{\text{CRS}}, p_{\text{CRS}}, g_{\text{CRS}}), A_{\text{CRS}}, B_{\text{CRS}}, C_{\text{CRS}})$ (this costs 4 exponentiations), also we need to compute $a = g^t$, $b = h^t$. The total number of exponentiations is 6 (2 exponentiations mod p, and 4 exponentiations mod p_{CRS}). The verifier needs to perform two times the verifier's algorithm for $\Pi_{\mathcal{DH}}$, one with the instance $((G_{\text{CRS}}, q_{\text{CRS}}, p_{\text{CRS}}, g_{\text{CRS}}), A_{\text{CRS}}, B_{\text{CRS}}, C_{\text{CRS}})$, the other one with the instance $((G, q, p, g), A, B, C)$, for a total amount of 4 exponentiations mod p_{CRS}, and 4 exponentiations mod p.

Case 2: Graph isomorphism.

- [Lin15]. We consider that the instance to be proved is composed by two graphs (G_0, G_1). Also we assume that to represent one graph with n vertices $k = n^2$ bits are necessary. In this case we remark that because the security parameter is $n = 1024$ we need to execute n times the protocol $\Pi_{\mathcal{GH}}$ described before. For the described assumptions we have that the first round of $\Pi_{\mathcal{GH}}$ is $P = \sigma(G_b)$ and $|P| = n^2$. Therefore the prover needs to run n executions of the DM commitment function to commit to P, where each of them costs 4 exponentiations. Also we need to execute n iteration of this process, for a total amount of $4n^2$ exponentiations mod p_{CRS}. Even in this case the verifier needs to checks if all opens with respect to the n commitments are correctly computed for a total amount of $4n^2$ exponentiations mod p_{CRS}.

- Our transform. In this case the prover $\mathcal{P}$ computes only 4 exponentiations mod p to compute the first round of $\Pi_{\mathcal{DH}}$. The verifier runs the verifier's algorithm of $\Pi_{\mathcal{DH}}$ and this requires 4 exponentiations mod p.

7.7 An Optimal-Sound (and Not Special Sound) 3-Round Perfect Special HVZK Proof

In this section we show a 3-round public-coin perfect special HVZK proof system that is optimal sound and not special sound. First of all we briefly describe the Σ-protocol of [MP03] to prove that, given a commitment `com` and a message m, m is committed in `com`. Then we show the protocol of [Vis06], that is a modification of [MP03], where given a commitment `com` and a value Ψ, allows to prove that the discrete logarithm of Ψ is committed in `com`.

In order to describe the protocol of [MP03] and [Vis06] we consider two prime p and q s.t. $p = 2q + 1$, a group of order $\mathcal{G}$ of order q such that the DDH assumption is hard. Also we consider two random elements, g and h, taken from $\mathcal{G}$.

We next describe Σ-protocol $\Pi_{Com} = (\mathcal{P}_{Com}, \mathcal{V}_{Com})$ of [MP03] for relation

$$\mathsf{Rel}_{Com} = \left\{ \left(\left((\mathcal{G}, q, g, h), v, \mathtt{com} = (\hat{g}, \hat{h}) \right), w \right) : \hat{g} = g^w, \hat{h} = h^{w+v} \right\}.$$

Common Input: $(\mathcal{G}, g, v, h, \mathtt{com} = (\hat{g}, \hat{h}), q)$ and relation Rel_{Com}.

Input of $\mathcal{P}_{Com}$: w s.t. $((\mathcal{G}, v, g, h, \mathtt{com} = (\hat{g}, \hat{h}), q), w) \in \mathsf{Rel}_{Com}$.

The protocol Π_{Com}:

1. The prover $\mathcal{P}_{Com}$ chooses r from $\mathcal{Z}_q$ and sends $(\widetilde{g} = g^r, \widetilde{h} = h^r)$ to $\mathcal{V}_{Com}$;
2. The verifier $\mathcal{V}_{Com}$ chooses a random challenge $e \leftarrow \mathcal{Z}_q$ and sends e to $\mathcal{P}_{Com}$;
3. $\mathcal{P}_{Com}$ sends $z = ew + r$ to $\mathcal{V}_{Com}$;
4. $\mathcal{V}_{Com}$ checks that $\hat{g}^e \widetilde{g} = g^z$ and $\left(\frac{\hat{h}}{h^v}\right)^e \widetilde{h} = h^z$ accepts if and only if the checks are successful.

In [Vis06] a similar protocol was used to prove that `com` is a commitment of the discrete logarithm of a value $\Psi \in \mathcal{G}$ with $h^\psi = \Psi$. Formally the protocol is for the $\mathcal{NP}$ language

$$L = \left\{ \left(\Psi = h^\psi, \mathtt{com} = (\hat{g} = g^w, \hat{h} = h^{w+\psi}) \right) : g, h \leftarrow \mathcal{G}, \psi \in \mathbb{Z}_q, w \in \mathbb{Z}_q \right\}$$

and for the corresponding relation

$$\mathsf{Rel}_L = \left\{ \left((\Psi = h^\psi, \mathtt{com} = (\hat{g} = g^w, \hat{h} = h^{w+\psi})), (w, \psi) \right) : g, h \leftarrow \mathcal{G}, \psi \in \mathbb{Z}_q, w \in \mathbb{Z}_q \right\}$$

The protocol follows Π_{Com} with the differences that the common input is $(\mathcal{G}, q, g, \Psi = h^\psi, h, \mathtt{com} = (\hat{g}, \hat{h}))$ and that the verifier decide whether to accept or not checking if it holds that $\hat{g}^e \widetilde{g} = g^z$ and $\left(\frac{\hat{h}}{\Psi}\right)^e \widetilde{h} = h^z$. While this protocol preserves the perfect special HVZK property, it is not a proof of knowledge for Rel_L neither special sound even though it still enjoys optimal soundness. We now proceed more formally.

Optimal soundness. We now consider an instance that is not in the $\mathcal{NP}$ language L, and show that, once the first round of the protocol is fixed, there exists only one challenge e s.t. the prover can answer successfully computing the third round z of the protocol. Consider the instance $\left(\Psi = h^{\psi}, \mathtt{com} = (\hat{g} = g^{w}, \hat{h} = h^{w+\psi\prime})\right) \notin L$ (with $\psi \neq \psi\prime$). Assume by contradiction that given the fist round of the protocol $(\widetilde{g}, \widetilde{h})$ there exist two distinct challenges e_0 and e_1 for which the prover can make the verifier accept with answers z_0, z_1 respectively. In the end we prove that $\psi = \psi\prime$.

Proof. Since the verifier accepts, it must be that for all $i \in \{0, 1\}$, the following checks are successful:

- $\hat{g}^{e_i}\widetilde{g} = g^{z_i}$;
- $\left(\frac{\hat{h}}{\Psi}\right)^{e_i}\widetilde{h} = h^{z_i}$.

It follows that $\hat{g}^{e_0-e_1} = g^{z_0-z_1}$ and $\left(\frac{\hat{h}}{\Psi}\right)^{e_0-e_1} = h^{z_0-z_1}$. Suppose that $h = g^{\omega}$, we get

$$g^{w\omega(e_0-e_1)} = \hat{g}^{(e_0-e_1)\omega} = g^{(z_0-z_1)\omega} = h^{(z_0-z_1)} = \left(\frac{\hat{h}}{\Psi}\right)^{e_0-e_1} = h^{z_0-z_1} = g^{\omega(w+\psi\prime-\psi)(e_0-e_1)}.$$

Therefore, if $e_0 \neq e_1$ we get the contradiction that $\psi = \psi\prime$. $\square$

Π_{Com} is not special sound for Rel_L. To argue that the protocol of [Vis06] is not special sound, we note that in order to compute a commitment $\mathtt{com}$ of the discrete logarithm of Ψ, knowledge of this discrete logarithm is not necessary since it is possible to compute $\mathtt{com} = (\hat{g}, h^{w} \cdot \Psi)$ with $w \in \mathbb{Z}_q$. Indeed, notice that the discrete logarithm ψ of Ψ is never used in the proof. Formally, we suppose that the protocol is special sound for the polynomial relation Rel_L and then construct an adversary $\mathcal{A}$ that, given $Y = g^{y} \in \mathcal{G}$, returns the discrete logarithm y of Y.

We have shown that there exist 3-round public-coin proof systems that are optimal sound and not special sound. It also easy to observe that special soundness implies optimal soundness.

Indeed, consider an $\mathcal{NP}$-Language L. All Σ-protocols for Rel_L must also be 3-round HVZK proofs for L with optimal soundness. If not, than the violation of optimal soundness ($\mathcal{P}^{\star}$ for a false statement can generate (a, c, z) and (a, c', z') with c' different from c and both accepting) implies directly also a violation of special soundness.

Bibliography

[ABB$^+$10] José Bacelar Almeida, Endre Bangerter, Manuel Barbosa, Stephan Krenn, Ahmad-Reza Sadeghi, and Thomas Schneider. A certifying compiler for zero-knowledge proofs of knowledge based on sigma-protocols. In *Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings*, volume 6345 of *Lecture Notes in Computer Science*, pages 151–167. Springer, 2010.

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 774–783. ACM, 2014.

[ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 535–548, 2013.

[ALSZ15] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 673–701, 2015.

[Bar02] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 345–355, 2002.

[BBC$^+$11] Pierre Baldi, Roberta Baronio, Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Countering GATTACA: efficient and secure testing of fully-sequenced human genomes. In *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 691–702, 2011.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 186–195, 2004.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[BDSG$^+$13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. *Why "Fiat-Shamir for Proofs" Lacks a Proof*, pages 182 – 201. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013/01/01/ 2013.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 103–112, 1988.

[BGJ$^+$17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal mpc. Cryptology ePrint Archive, Report 2017/1088, 2017. https://eprint.iacr.org/2017/1088.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513, 1990.

[BPSV08] Carlo Blundo, Giuseppe Persiano, Ahmad-Reza Sadeghi, and Ivan Visconti. Improved security notions and protocols for non-transferable identification. In *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, pages 364–378, 2008.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 626–643, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 424–441. Springer, 1998.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In YvoG. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Berlin Heidelberg, 1994.

[CDV06] Dario Catalano, Yevgeniy Dodis, and Ivan Visconti. Mercurial commitments: Minimal assumptions and efficient constructions. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 120–144, 2006.

[CG15] Pyrros Chaidos and Jens Groth. Making sigma-protocols non-interactive without random oracles. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 650–670, 2015.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 209–218, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 78–96. Springer Berlin Heidelberg, 2006.

[CLP13] Ran Canetti, Huijia Lin, and Omer Paneth. Public-coin concurrent zero-knowledge in the global hash model. In *TCC*, pages 80–99, 2013.

[CO18] Michele Ciampi and Claudio Orlandi. Combining private set-intersection with secure two-party computation. Cryptology ePrint Archive, Report 2018/105, 2018. https://eprint.iacr.org/2018/105.

[COSV16] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 270–299. Springer, 2016. Full version http://eprint.iacr.org/2016/566.

[COSV17a] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 711–742. Springer, 2017. Full version http://eprint.iacr.org/2017/931.

[COSV17b] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Four-round concurrent non-malleable commitments from one-way functions. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 127–157, 2017. Full version http://eprint.iacr.org/2016/621.

[COSV17c] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Round-optimal secure two-party computation from trapdoor permutations. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 678–710. Springer, 2017.

[CPS13] Kai-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 231–240. ACM, 2013.

[CPS$^+$16a] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved or-composition of sigma-protocols. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 112–141. Springer, 2016. Full version http://eprint.iacr.org/2015/810.

[CPS$^+$16b] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline OR composition of sigma protocols. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 63–92. Springer, 2016. Full version http://eprint.iacr.org/2016/175.

[CPSV16] Michele Ciampi, Giuseppe Persiano, Luisa Siniscalchi, and Ivan Visconti. A transform for NIZK almost as efficient and general as the fiat-shamir transform without programmable random oracles. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 83–111. Springer, 2016.

[CT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers*, pages 143–159, 2010.

[CT12] Emiliano De Cristofaro and Gene Tsudik. Experimenting with fast private set intersection. In *Trust and Trustworthy Computing - 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings*, pages 55–73, 2012.

[CV05] Dario Catalano and Ivan Visconti. Hybrid trapdoor commitments and their applications. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 298–310, 2005.

[CV07] Dario Catalano and Ivan Visconti. Hybrid commitments and their applications to zero-knowledge proof systems. *Theor. Comput. Sci.*, 374(1-3):229–260, 2007.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430. Springer, 2000.

[Dam10] Ivan Damgård. On Σ-protocol. http://www.cs.au.dk/~ivan/Sigma.pdf, 2010.

[DCV05] Giovanni Di Crescenzo and Ivan Visconti. Concurrent zero knowledge in the public-key model. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 816–827. Springer, 2005.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 542–552, 1991.

[DFN06] Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC, 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 41–59, 2006.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 426–437, 2003.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22, 1976.

[DMP87] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, pages 52–72, 1987.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 283–293, 2000.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 581–596, 2002.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[DNT12] Morten Dahl, Chao Ning, and Tomas Toft. On secure two-party integer division. In *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, pages 164–178, 2012.

[DSDCPY94] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of SZK. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 454–465, 1994.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.*, pages 205–210. Plenum Press, New York, 1982.

[Fei90] Uriel Feige. Alternative models for zero knowledge interactive proofs. Master's thesis, Weizmann Institute of Science, Rehovot, Israel, 1990. Ph.D. thesis.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 303–324, 2005.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 60–79, 2012.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 308–317. IEEE Computer Society, 1990.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM J. on Computing*, 29(1):1–28, 1999.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 1–19, 2004.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 526–544, 1989.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, pages 416–426, New York, NY, USA, 1990. ACM.

[GK96a] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology*, 9(3):167–190, 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 102–113, 2003.

[GKM$^+$00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 325–335, 2000.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32, 1989.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 51–60, 2012.

[GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 448–476. Springer, 2016.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 174–187, 1986.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[GMY06a] Juan A. Garay, Philip MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, 2006.

[GMY06b] Juan A. Garay, Philip MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, 2006.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2006.

[GOSV14] Vipul Goyal, Rafail Ostrovsky, Alessandra Scafuro, and Ivan Visconti. Black-box non-black-box zero knowledge. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 515–524. ACM, 2014.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 695–704, 2011.

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 1128–1141, 2016. Full version: Cryptology ePrint Archive, Report 2015/1178.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer, 1988.

[Gro04] Jens Groth. *Honest verifier zero-knowledge arguments applied.* Dissertation Series DS-04-3, BRICS. PhD thesis. xii+119 pp, 2004.

[GRRV14] Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. An algebraic approach to non-malleability. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 41–50, 2014.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 415–432, 2008.

[GSV07] Juan A. Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, pages 330–342, 2007.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[HHPV17] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. Cryptology ePrint Archive, Report 2017/1056, 2017. https://eprint.iacr.org/2017/1056.

[HKR$^+$14] Feng Hao, Matthew Nicolas Kreeger, Brian Randell, Dylan Clarke, Siamak Fayyaz Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in large-scale electronic voting. *USENIX Journal of Election Technology and Systems*, 2(3), July 2014.

[HL08] Carmit Hazay and Yehuda Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, pages 155–175, 2008.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions.* Information Security and Cryptography. Springer, 2010.

[HOS17] Per Hallgren, Claudio Orlandi, and Andrei Sabelfeld. Privatepool: Privacy-preserving ridesharing. In *IEEE 30th Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pages 276–291, 2017.

[HRVW09] Iftach Haitner, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Inaccessible entropy. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 611–620. ACM, 2009.

[HV17] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I*, pages 175–203, 2017.

[IKN$^+$17] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. Cryptology ePrint Archive, Report 2017/738, 2017. http://eprint.iacr.org/2017/738.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 145–161, 2003.

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 575–594, 2007.

[JL10] Stanislaw Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, pages 418–435, 2010.

[Khu17] Dakshita Khurana. Round optimal concurrent non-malleability from polynomial hardness. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 139–171. Springer, 2017.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 54–70, 2013.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 818–829, 2016.

[KMRS14] Seny Kamara, Payman Mohassel, Mariana Raykova, and Seyed Saeed Sadeghian. Scaling private set intersection to billion-element sets. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, pages 195–215, 2014.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 335–354, 2004.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 724–741, 2015.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of fiat-shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 224–251. Springer, 2017.

[Lin10] Yehuda Lindell. Foundations of cryptography 89-856. http://u.cs.biu.ac.il/~lindell/89-856/complete-89-856.pdf, 2010.

[Lin14] Yehuda Lindell. An efficient transform from Sigma Protocols to NIZK with a CRS and non-programmable random oracle. Cryptology ePrint Archive, Report 2014/710, 2014. http://eprint.iacr.org/2014/710/20150906:203011.

[Lin15] Yehuda Lindell. An efficient transform from Sigma protocols to NIZK with a CRS and non-programmable random oracle. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, pages 93–109, 2015.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of yao's protocol for two-party computation. *J. Cryptology*, 22(2):161–188, 2009.

[LP11] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 705–714. ACM, 2011.

[LP15] Huijia Lin and Rafael Pass. Constant-round nonmalleable commitments from any one-way function. *J. ACM*, 62(1):5:1–5:30, 2015.

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, volume 4948 of *Lecture Notes in Computer Science*, pages 571–588. Springer, 2008.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In *Proceedings of the 41st Annual ACM Symposium on Theory of*

Computing,STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009, pages 179–188, 2009.

[LS90] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *Advances in Cryptology - CRYPTO*, 1990.

[LT13] Helger Lipmaa and Tomas Toft. Secure equality and greater-than tests with sub-linear online complexity. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 645–656, 2013.

[M13] Djekic M. Cryptography of the ancient sparta. 2013. www.australianscience.com.au/technology/a-scytale-cryptography-of-the-ancient-sparta/ A Scytale.

[Mau15] Ueli Maurer. Zero-knowledge proofs of knowledge for group homomorphisms. *Designs, Codes and Cryptography*, pages 1–14, 2015.

[Mea86] Catherine A. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 7-9, 1986*, pages 134–137, 1986.

[MN12] Payman Mohassel and Salman Niksefat. Oblivious decision programs from oblivious transfer: Efficient reductions. In *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, pages 269–284, 2012.

[MP03] Daniele Micciancio and Erez Petrank. Simulatable commitments and efficient concurrent zero-knowledge. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 140–159, 2003.

[MP12] Mohammad Mahmoody and Rafael Pass. The curious case of non-interactive commitments - on the power of black-box vs. non-black-box use of primitives. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 701–718. Springer, 2012.

[MV16] Arno Mittelbach and Daniele Venturi. Fiat-shamir for highly sound protocols is instantiable. In Vassilis Zikas and Roberto De Prisco, editors, *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings*, volume 9841 of *Lecture Notes in Computer Science*, pages 198–215. Springer, 2016.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NMH$^+$10] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. Botgrep: Finding P2P bots with structured graph analysis. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, pages 95–110, 2010.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437, 1990.

[OOS17] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-n OT extension with application to private set intersection. In *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, pages 381–396, 2017.

[OPV08] Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti. Constant-round concurrent non-malleable zero knowledge in the bare public-key model. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pages 548–559, 2008.

[OPV10] Rafail Ostrovsky, Omkant Pandey, and Ivan Visconti. Efficiency preserving transformations for concurrent non-malleable zero knowledge. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2010.

[ORS15] Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2015.

[ORSV13] Rafail Ostrovsky, Vanishree Rao, Alessandra Scafuro, and Ivan Visconti. Revisiting lower and upper bounds for selective decommitments. In *TCC*, pages 559–578, 2013.

[OV12] Rafail Ostrovsky and Ivan Visconti. Simultaneous resettability from collision resistance. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:164, 2012.

[Pas03a] Rafael Pass. On deniability in the common reference string and random oracle model. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 316–337, 2003.

[Pas03b] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2003.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 232–241. ACM, 2004.

[Pas13] Rafael Pass. Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In *TCC*, pages 334–354, 2013.

[Pol16] Antigoni Polychroniadou. *On the Communication and Round Complexity of Secure Computation*. PhD thesis, Aarhus University, December 2016.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 57–74, 2008.

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 404–413. IEEE Computer Society, 2003.

[PR05a] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 563–572, 2005.

[PR05b] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 533–542. ACM, 2005.

[PR05c] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 533–542, 2005.

[PR08a] Rafael Pass and Alon Rosen. Concurrent nonmalleable commitments. *SIAM J. Comput.*, 37(6):1891–1925, 2008.

[PR08b] Rafael Pass and Alon Rosen. New and improved constructions of nonmalleable cryptographic protocols. *SIAM J. Comput.*, 38(2):702–752, 2008.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pages 515–530, 2015.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 797–812, 2014.

[PSZ16] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. Cryptology ePrint Archive, Report 2016/930, 2016. http://eprint.iacr.org/2016/930.

[PW10] Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, pages 638–655, 2010.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394, 1990.

[RR17] Peter Rindal and Mike Rosulek. Improved private set intersection against malicious adversaries. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, pages 235–259, 2017.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 543–553. IEEE Computer Society, 1999.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer New York, 1989.

[SCO$^+$01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598. Springer, 2001.

[Sha80] Adi Shamir. On the power of commutativity in cryptography. In *Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherland, July 14-18, 1980, Proceedings*, pages 582–595, 1980.

[SV12] Alessandra Scafuro and Ivan Visconti. On round-optimal zero knowledge in the bare public-key model. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 153–171. Springer, 2012.

[T$^+$07] Tomas Toft et al. Primitives and applications for multi-party computation. *PhD Thesis, University of Aarhus, Denmark*, 2007.

[TLP$^+$17] Sandeep Tamrakar, Jian Liu, Andrew Paverd, Jan-Erik Ekberg, Benny Pinkas, and N. Asokan. The circle game: Scalable private membership test using trusted hardware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pages 31–44, 2017.

[Vis06] Ivan Visconti. Efficient zero knowledge on the internet. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 22–33, 2006.

[VV09] Carmine Ventre and Ivan Visconti. Co-sound zero-knowledge with public keys. In *Progress in Cryptology - AFRICACRYPT 2009, Second International Conference* *on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, volume 5580 of *Lecture Notes in Computer Science*, pages 287–304. Springer, 2009.

[Wee09] Hoeteck Wee. Zero knowledge in the random oracle model, revisited. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 417–434, 2009.

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 531–540. IEEE Computer Society, 2010.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.

[YZ06] Moti Yung and Yunlei Zhao. Interactive zero-knowledge with restricted random oracles. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 21–40, 2006.

[YZ07] Moti Yung and Yunlei Zhao. Generic and practical resettable zero-knowledge in the bare public-key model. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 129–147. Springer, 2007.