Università degli Studi di Salerno

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

Dottorato di Ricerca in Ingegneria dell'Informazione
Ciclo 35

TESI DI DOTTORATO / PH.D. THESIS

Provable Security: the Good, the Bad, and the Ugly

GENNARO AVITABILE

SUPERVISOR: **PROF. IVAN VISCONTI**

PHD PROGRAM DIRECTOR: **PROF. PASQUALE CHIACCHIO**

Anno 2023

Abstract

Since modern cryptography was born around the late 1970s, a myriad of cryptographic constructions and protocols have been proposed. The field quickly developed into a science whose results have had great impact on people's lives. Some examples are secure communication over the internet, distributed digital currencies, electronic elections, and more. Such a progress was boosted by the diffusion of a methodology called the *provable security* paradigm. It provides a precise framework to formalize and prove the security of a cryptographic construction. Provable security is based on three pillars: (i) definitions, (ii) assumptions, and (iii) proofs. The definition states when the system can be considered secure and what are the capabilities of an adversary attacking the construction. The proof demonstrates that the construction satisfies the definition, assuming that all the assumptions hold. Provable security provides objective ways to compare different constructions, as well as more reassurances on their security. However, it is not devoid of pitfalls. For example, a definition might not model the real world correctly, and thus any proof that a construction satisfies such definition would be worthless in practice. Furthermore, it might happen that security proofs containing errors will not get detected because of the complexity (or oversimplification) of the proof itself.

This thesis explores such multifaceted nature of provable security through two parts. In the first part, we focus on the recent development of automatic contact tracing systems (ACTs). When the COVID-19 pandemic hit, automatic contact tracing was proposed as an effective way to slow the spread of the virus down by detecting likely infected people earlier with the help of technology. Citizens would use a smartphone app, and users at risk of being infected - as

they were in proximity of an infected individual - would be notified by the smartphone. Due to the widespread adoption that was expected for ACTs, privacy and integrity were both key concerns.

The DP3T team proposed an ACT [114] which was shortly after implemented and deployed over smartphones by Apple and Google with the name of GAEN. Informal security assessments were performed by the DP3T team, including wrong or misleading claims about the privacy and integrity guarantees that ACTs could provide. Several attacks to DP3T pointed out by other researchers were deemed as inherent by analyses that considered very powerful adversaries. However, the concrete attacks could have been carried out by much weaker adversaries to which other ACTs could have possibly resisted. We model these and novel integrity and privacy attacks with a focus on mass surveillance and analyze the security of DP3T w.r.t. them. We propose two new ACTs named Pronto-C2 and Pronto-B2, which encompass DP3T/GAEN both in terms of privacy and integrity guarantees. Our ACTs also demonstrate that such attacks are not inherent. Finally, we consider the terrorist attack conjectured by Vaudenay [116]. It involves a malicious party (i.e., the terrorist) bribing infected users to inject false alerts in the ACT. We show how to concretely implement automated terrorist attacks to jeopardize the integrity of GAEN.

In the second part of this thesis, we provide novel contributions in the area of threshold cryptography. In particular, we focus on proofs over threshold relations, threshold ring signatures, and extendable threshold ring signatures. We point out several fallacies in the usage of the provable security paradigm in prior works published at major cryptography conferences [5, 64]. Such issues include errors in the security proofs as well as inadequate definitions where the real-world system's requirements and adversary's capabilities are not matched by the definition. We overcome such issues proposing stronger definitions, new constructions, and revisited security proofs. Additionally, our new constructions improve the previous ones in terms of efficiency, security, and/or features.

Contents

Chapter 1

Introduction

Historically, cryptography was seen as the art of conceiving or solving intricate codes. Either military or dissidents used these codes to communicate in a way that prevented their enemies to understand what was being said, even if they intercepted the messages. In those times, cryptography heavily relied upon creativity, with little to no underlying theory, and there was no objective definition of what a "good" code was supposed to be.

During the late 1970s, there was a revolution in the world of cryptography, known as the birth of *modern cryptography.* Theory started to be developed, transforming cryptography to a science based on strong and rigorous mathematical foundations. Nowadays, cryptography is ubiquitous. It is at the hearth of methods to ensure integrity, to perform secure distributed computation, to run electronic elections, to build digital currencies, and much more.

This huge progress was made possible by the introduction of a rigorous method of building and evaluating cryptographic constructions known as the *provable security* paradigm.

Originally, cryptographic constructions used to be evaluated for their apparent complexity. The analysis of the schemes consisted of evaluations of possible attacks and if no attack was found, then the construction was considered secure. On the other hand, if an attack was found the scheme was fixed to thwart that attack, and such process could be repeated multiple times.

The provable security paradigm strongly refutes this approach: instead of a process to agree that a construction is broken, as may be highlighted by successful attacks, its purpose is to formally define what is secure and to be able to prove its security. Provable security achieves this by the means of three principles:

- **Definitions**: A *formal* definition should clearly state what does it mean for that particular cryptographic construction to be secure.

- **Assumptions**: The vast majority of cryptographic constructions is not proven secure unconditionally. Instead, the proofs usually rely on *unproven* assumptions, typically involving some mathematical problem that is conjectured to be hard to solve. It is crucial to explicitly state and clarify what are the assumptions being used.

- **Proofs**: It is a rigorous mathematical proof that a cryptographic construction satisfies the definition, provided that the used assumptions hold.

In the following, a more detailed overview on the role of these three principles of the provable security paradigm is given.

Definitions. The formulation of a proper security definition consists in carefully addressing two objectives:

- **Fixing a security goal**: it can be expressed from the point of view of the users as a desired security guarantee or, conversely, from the point of view of the adversary, as what constitutes a successful attack. It should use a mathematical formalism in order to eliminate any form of ambiguity.

- **Fixing a threath model**: the threath model specifies what is the power of the adversary. In particular, it should specify all the capabilities the adversary has (e.g., access to the private state of a subset of corrupted parties, its computational power, etc.), without specifying any attack strategy. Indeed, security definitions should be quantified on all the adversaries having a certain set of capabilities since it is impossible to predict how certain capabilities will be compiled into a real-life attack strategy. The most commonly considered adversary is the probabilistic polynomial time (PPT) algorithm. Namely, an algorithm which has access to a random tape and runs in polynomial time[1]. Additional capabilities are usually modeled as oracles in the security definition. For example, to model corruption, the adversary could be given access to a corruption oracle that gives him the private state of a certain number of users (e.g., up to half of them).

Stating a precise definition gives at least other two significant advantages. First, it reveals the essential aspects of the problem under examination while highlighting the subtleties that may not be so evident at a first sight. Second, it gives a more objective tool to compare different constructions: for example a less efficient (e.g., in terms of size or time complexity) construction may be preferred to a more efficient one because it satisfies a stronger definition.

Assumptions. Cryptographic constructions cannot usually be proven secure unconditionally, instead they rely on mathematical assumptions. Such assumptions are only conjectured to be true, and have to be stated precisely so that they are easy to (eventually) confute. During the last decades, several assumptions have been proposed and some of them are still believed to hold (e.g., the DDH assumption or the RSA factoring assumption). One key factor when comparing two constructions that provide the same guarantees is to look at their assumptions. In this case, a construction is generally preferred to another if it uses a better-studied assumption, or if it uses a strictly weaker assumption than the other one (i.e., its assumption is implied by the other one).

Proofs. A proof that a cryptographic construction satisfies a certain definition under certain assumptions is usually given via a reduction:

[1]Other examples are quantum polynomial time adversaries, which are adversaries that run in polynomial time but have access to a quantum model of computation, or even unbounded time adversaries. In the latter case, security is proven using information-theoretic arguments.

one first assumes that there exists an adversary breaking the construction, and then uses such adversary to build an algorithm that breaks the assumption. Since the assumption is considered to be true, the contradiction that has been reached guarantees the security of the construction itself.

Idealized models. Whenever a construction can be proved secure only using standard assumptions that reduce to a mathematical problem, it is said that such construction can be proved secure in the *plain model*. There are other interesting alternative models that introduce new heuristic assumptions but that enable to prove security of enhanced primitives. Prominent examples are the random oracle model (ROM), and the common reference string (CRS) model.

Random Oracle Model. The random oracle model is an idealized model in which the existence of a random oracle which returns a uniformly random value whenever it is queried on a new input is assumed. When a primitive proven secure in the ROM is concretely instantiated, the random oracle is heuristically replaced by a cryptographic hash function. Although there are theoretical examples showing the random oracle heuristic to fail [32], no weaknesses are known for "natural usages" of the random oracle.

CRS model. The common reference string (CRS) model assumes that all the parties involved are given access to the same string that is generated from a fixed distribution. Such string is considered to be honestly generated via a so-called trusted setup. Therefore, schemes proven secure in the CRS model are only secure given that the setup was correctly performed.

1.1 Provable Security: the Good, the Bad, and the Ugly

Although it is undoubted that the introduction of provable security has been extremely beneficial to the development of cryptography as we know it today, it is not devoid of pitfalls and dangers. In this section, we describe the multifaceted nature of provable security with the help of three adjectives: the Good, the Bad, and the Ugly.

The Good. The vast literature in the area of modern cryptography demonstrates the great impact provable security has had. Since the birth of modern cryptography, researchers have designed a wide range of cryptographic primitives offering many different functionalities. Among those, encryption, digital signatures, cryptographic hashing, secure multi-party computation, zero-knowledge proofs, and much more.

The Bad. Single cryptographic primitives in isolation are often not enough to solve the challenges particular application scenarios involve. To tackle these challenging problems, more complex systems are usually built combining many cryptographic building blocks together in order to provide a complex functionality having both integrity and privacy constraints. This means that, aside providing its intended functionality under normal circumstances, the system is expected to have a certain level of resilience in presence of users or external entities with adversarial behaviour. Additionally, it should simultaneously protect the information which is deemed confidential, resulting in what is commonly referred to as a *secure* system. By using the world *secure*, one means that the system provides the intended integrity and privacy guarantees. However, in many cases the meaning of this word is rather vague. Either the requirements themselves are ambiguously formulated only by the means of natural language, or some requirements are declared as fulfilled just because some cryptographic primitive is employed, perhaps incorrectly. This approach avoids the usage of a rigorous methodology like the provable security paradigm. As a result, one might be tempted to justify the security of the system making informal claims that can be either misleading or incorrect since they are not formulated within a rigorous framework.

The Ugly. It is important to notice that provable security still does not put an end to attacks. Excluding implementation errors that can afflict any software system, cryptographic constructions could be successfully attacked even if they have a security proof for several reasons, among those:

- **Inadequate definitions:** The security guarantee of the definition does not match the requirement of the system, or the threat model underestimates the capabilities of the real-world adversary. This means that when formulating a definition one has to check whether it is close enough to the real world or not.
- **Problematic proofs:** As the complexity of a system/primitive increases, usually proofs become more involved and difficult to check. This posits the risk that constructions believed to be secure and even used in practice can be discovered to be broken at a much later point. While experience is often a great tool when evaluating the quality of a security proof, it could also be deceiving. Whenever a scheme seems sound and the arguments given in the proofs resemble others from past experience, a reviewer could be tempted to skim on the security reductions without carefully checking all the details.
- **Assumptions break:** An assumption on which the proof was based on turns out to be false. In this case, the proof becomes worthless. A very recent example is the Supersingular Isogeny Diffie Hellman (SIDH) problem. In summer 2022, it was found to be solvable on a laptop for parameters choice that were aiming at security level 1 of the Post-Quantum Cryptography standardization process currently ran by NIST [36, 87].

A way more comprehensive discussion on the ugly side of provable security can be found in [81]. In this position paper, Koblitz and Menezes deeply discuss these and other concerns, including the usage of idealized models and tightness issues.

1.2 Our Contributions

This thesis explores such multifaceted nature of provable security. The contributions of this thesis are related to two different areas, namely automatic contact tracing and threshold cryptography. In the first part of the thesis (cfr., Chap. 3), we use the recent development of automatic contact tracing systems (ACTs) to showcase the *bad* consequences that neglecting provable security can bring when constructing complex systems with privacy and integrity constraints.

We show concrete attacks to the most adopted ACT (i.e., DP3T/GAEN) regarding both integrity and privacy. On the positive side, we propose new ACTs which disprove some DP3T's claims about the inherent nature of some integrity and privacy issues in ACTs.

In the second part of the thesis (cfr., Chap. 4), we provide new contributions in the area of threshold cryptography. In particular, we propose new compact zero-knowledge proofs for threshold relations, new threshold ring signatures, and new extendable threshold ring signatures (ETRS) with enhanced anonymity properties.

While confronting ourselves with previous works we improve on in the area of threshold cryptography, we analyse some concrete examples of the *ugly* side of provable security such as inadequate definitions or fallacies in the security reductions.

Co-authored works related to this thesis. Parts of the results of this thesis are also contained in co-authored articles [10, 11, 13, 14, 16]. All the co-authors gave their consent to include such results in this thesis. The thesis includes parts of text that are verbatim reported from such co-authored works. For the seek of readability, we will omit quotation marks when reporting such text.

1.3 First Contribution: Automatic Contact Tracing

Part of the results in the area of ACTs included in this thesis were published in peer reviewed conferences and journals such as Workshop on Secure IT Technologies against COVID-19 (CoronaDef, co-located with NDSS 2021) [13], IEEE Internet Computing [14], and ACNS 2021 [16]. In particular, parts of the results published in [13, 14] are also present in the PhD thesis[2] of Vincenzo Botta who was a co-author of such publications. The results of [13, 14] are part of a full version that was published on ePrint [12]. The results of [16] are part of a full version that was published on ePrint [15].

Contact tracing. At the beginning of 2020, a new virus named Sars-Cov-2 was already spreading in Wuhan, China. Soon it became very clear that a new pandemic had begun.

Governments around the world put in place strong measures trying to contain the virus. Lockdowns were established, basically forcing people to stay home at all times. Nevertheless, the virus had already started to quickly spread, hospitals were saturated and many people were dying every day. In this climate of terror and fear, everyone was looking for solutions that could alleviate the situation. People wanted their freedom back again as soon as possible, furthermore the economic impact of this virus was considerable, with many lost jobs. One key strategy used by health authorities to limit the spread of the virus is contact tracing. When a person is tested positive, she is interviewed by a contact tracer to determine all her recent close contacts in the contagious time window (e.g., 10 days before the beginning of the symptoms). Then, all the people recently met by the infected citizen are informed, tested, and proper restrictive actions are taken.

Automatic contact tracing. A major problem with COVID-19 is that the virus spreads very quickly while the procedures to detect infected people and to find and inform potentially infected people are slow. When a new infected person is detected, too much time is spent to inform her recent contacts and to take proper restrictive actions. Commonly, when a new infected person is discovered, by the time her recent contacts are informed they have already had a significant chance to infect others.

In order to improve on this situation, many researchers proposed automatic systems for contact tracing (ACTs, for short) [59]. The hope was that such systems could dramatically increase chances that

[2]The thesis will be made available at http://elea.unisa.it/handle/10556/50.

recent contacts of an infected person were informed before infecting others. Ideally, whenever a person is tested positive, all her recent contacts (i.e., persons that have been in close proximity to the infected one) should be immediately notified. This would allow to promptly take appropriate countermeasures.

Privacy threats of automatic contact tracing. In 2013, Edward Snowden disclosed global surveillance programs [46] opening a world-wide discussion about the trade-off between individual privacy and collective security. Motivated by such risks, several researchers and institutions have proposed and advertised to citizens the possibility of realizing automatic contact tracing systems that also preserve privacy to some extent.

Integrity threats of automatic contact tracing. Given the large-scale deployment that is expected for ACTs to effectively mitigate the spread of the virus [59], the integrity of such systems is another key concern. In a nutshell, one would want from ACTs a certain resilience to adversarial attacks. For example, users should only receive a notification if they actually met an infected person, and not because a malicious party targeted them. Notice that a notification from the ACT has concrete consequences in the physical world, potentially putting notified users in quarantine, or making the health system perform useless tests.

1.3.1 The Response of the Scientific Community

Motivated by the perceived need of designing and developing privacy-preserving ACTs as soon as possible, several projects involving researchers and experts from various countries born with the aim of proposing new systems.

In Europe, the Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) project [6] was launched as a Swiss-based organization with "more than 130 members across eight European countries". It incubated various candidate systems among which a European ACT should have been selected to be developed. Similar initiatives also born in the United States such as Private Automated Contact Tracing (MIT-PACT, for short) [94], or a similar project from the University of Washington UW-PACT [37]. Given their widespread adoption, smartphones were immediately considered as devices for the users of ACTs. Bluetooth Low Energy (BLE) was preferred as communication channel as it is not *directly* correlated to location information and can work without relying on any base station.

The BLE-based approach. BLE-based ACTs commonly rely on the use of pseudonyms that smartphones announce through BLE identifier beacons. To prevent easy tracking attacks, after a short period of time called time slot or epoch (e.g., 15 minutes), each smartphone replaces the already announced pseudonym with a (seemingly independent) new one. Each smartphone receives pseudonyms sent by others and stores them locally. Therefore, a smartphone will have a database of the announced pseudonyms and a database of the received pseudonyms. The main idea is that whenever a person tests positive, smartphones that have been physically close to the smartphone of the infected person should be notified.

In order to realize this, the smartphone of the infected person should use the above two databases to somehow reach out the smartphones that have recently been physically close to it. This communication is achieved through a backend server as follows. First, the smartphone of the infected person will use the above two databases to communicate data to the backend server. The server could then run some computations on data received from smartphones of infected citizens. The server will also use collected/computed data to answer queries of smartphones that desire to check if there is any notification for them.

Centralized vs decentralized ACTs. An important point of the design of a BLE-based ACTs is the generation of pseudonyms used by smartphones. Two major approaches were proposed. In the centralized approach, pseudonyms are generated by the server. Each smartphone, during the setup of the ACT smartphone application, connects to the server and receives its pseudonyms. Therefore, the server knows all the pseudonyms honestly used in the system. This is a clear open door to mass surveillance. Such dangers are discussed in [114]. The centralized approach is part of the protocols named NTK and ROBERT that were developed inside the PEPP-PT initiative [6].

The decentralized approach instead breaks the obvious link among pseudonyms belonging to the same smartphone by letting the smartphone itself generate such pseudonyms. However, it also introduces privacy and integrity threats that are not an issue in centralized systems. The different views on the paradigm to be adopted caused a split in PEPP-PT and led to the birth of another project called Decentralized Privacy-Preserving Proximity Tracing (DP3T, for short)[3] [114].

1.3.2 The Move of Apple and Google

The unlinkability of pseudonyms advertised in BLE identifier beacons is useless if the BLE MAC address associated to a smartphone does not change in a synchronized way with the pseudonyms. Indeed, if the MAC address remains fixed while the advertised pseudonym changes, it is straightforward to deduce that two different pseudonyms associated to the same MAC address come from the same device [20]. The vast majority of smartphones has either iOS or Android as operating system. Both iOS and Android have strict restrictions on updating the BLE MAC address. In contrast, the contact tracing application should be able to work in background and should have full control over the BLE MAC address.

As a result, it is problematic to realize BLE-based privacy-preserving contact tracing applications without help from Apple and Google. Interestingly, Apple and Google have released updates of iOS and Android called Google Apple Exposure Notification (GAEN, for short) [68]. GAEN provides the part concerning the generation, rotation, exposure, and storage of pseudonyms, thus solving the MAC address linkability problem. Access to GAEN is given exclusively to applications endorsed by governments and approved by Apple and Google.

However, instead of providing a general API, GAEN forces a specific contact tracing protocol, which is almost identical to DP3T. If one wants to implement a smartphone application that needs to rotate the

[3]Documents and code of this project can be found at https://github.com/DP-3T.

BLE MAC address synchronously with the content of the BLE identifier beacon[4], then he must use GAEN and therefore he is stuck with its approach for pseudonyms generation and exposition (i.e., the one of DP3T). This lack of flexibility makes the centralized approach not implementable, since it relies on pseudonyms generated by the server and then advertised in the BLE identifier beacon by the smartphone. On the other hand, the generation of pseudonyms can only happen inside the smartphone when using GAEN. This decision made by Apple and Google basically forced countries to switch to the decentralized model, even if they initially preferred the centralized one[5]. Many countries realized ACTs based on GAEN. They developed smartphone applications that are essentially wrappers of GAEN. Their main function is to set the parameters in GAEN function calls and to provide a user interface. Additionally, they provide an upload authorization mechanism and a server to gather and spread infected users' pseudonyms for which Google also provides a reference implementation [66]. GAEN-based contact tracing systems are still widely used in Europe (e.g., Austria, Belgium, Germany, Ireland, Italy, Poland, Spain, Switzerland, and more). Moreover, in the US, several states have adopted GAEN-based systems.

1.3.3 Security Issues of DP3T/GAEN

Since the first proposals of ACTs came out, researchers started to investigate their security. Many of these works focused on the analysis of DP3T [114]. Due to its great similarity with GAEN, practically all the results about it also apply to GAEN-based ACTs. Researchers elucidated both privacy and integrity issues in DP3T/GAEN. For a more detailed overview, see Sec. 3.2.

Privacy threats. Regarding privacy attacks, the most concerning one is the *paparazzi* attack by Vaudenay [115]. In his work, Vaudenay

[4]Notice that operating systems also restrict BLE usage in background. Therefore, even ignoring the MAC address problem, applications not using GAEN would drain the smartphone's battery much faster.

[5]See the case of UK that tried to develop a system without GAEN but had to give up https://www.bbc.com/news/technology-53095336.

presented the paparazzi attack as a de-anonymization attack, however it can also be seen as a tracking attack. Basically, in DP3T it is very easy for an attacker to track the locations visited by the infected users during the contagious time window. The attacker just needs to collect users' pseudonyms using an undetectable passive antenna (i.e., working only in reception mode). Then, when a user becomes infected, the attacker is able to link all the pseudonyms she used during the contagious time window (and thus to infer the locations the user visited during that time).

Integrity threats. In the context of ACTs, preserving the integrity of the system means preventing false alerts. A concerning attack allowing to direct false positive alerts to specific targets has been pointed out in [96, 115]. Indeed, GAEN-based contact tracing systems can be abused through replay attacks. In this case, the pseudonyms sent by an individual considered at risk (e.g., a person who is taking a test) are transmitted by an adversary to a different location in order to create fake proximity contacts. Another class of attacks leading to false alerts involves bribing. Vaudenay conjectured that decentralized ACTs like DP3T/GAEN could have been vulnerable to what he called the *terrorist* attack [116]. The terrorist attack involves users reporting pseudonyms that differ from the ones used during the previous days. Then, in principle, there could be smart contracts automatically rewarding such users. This black market could lead to a massive amount of fake notifications without relying on replay attacks.

1.3.4 Provable Security and Automatic Contact Tracing

Given all the attacks to GAEN/DP3T that have been demonstrated in the literature, a natural question that follows is whether the principles of provable security were applied in designing DP3T: the answer is no. One might wonder why provable security was not taken into consideration. Although it is impossible to give a definitive answer to this question, some of the reasons may include the following:

- There was basically no literature on the topic. There was no clear understanding of what where the requirements for privacy-preserving contact tracing systems.

- There was a climate of fear and urgency. People wanted solutions as soon as possible. Stating that many months of work would be needed only to figure out precise security goals and threat models could have been unpopular.
- Scientific discussion became polarized. Even though there were forum boards to discuss the projects, it was believed that everyone pointing out problems wanted to get in the way of the project for personal reasons.

Later on, independent works made the effort to formulate rigorous security models and to analyse ACTs in the provable security paradigm [34, 50] confirming vulnerabilities of DP3T or more clearly elucidating its security guarantees.

1.3.5 Security Assessments by the DP3T Team

Although outside the rigorous approach of provable security, the DP3T team performed a security assessment comparing the ACTs approaches that were known at the time [110]. The document explores both privacy and integrity risks.

Privacy risks. The document [110] first explores what they call inherent risks (IR) of ACTs. In particular, in IR1 it is said that "any proximity tracing system that notifies users that they are at risk enables a motivated attacker to identify the infected people that he or she has been physically near. This risk is a consequence of the basic proximity tracing functionality.". Then, they provide some examples such as the fact that whenever a notified user only met one person in the contagious time window, then the user is able to de-anonymize such infected user. Although such attack vectors are inherent, when facing these situations one should factor out inherent leaks by requiring that the system leaks nothing more than what it is inherently leaked by its functionality. However, these examples were used to argue that all sort of de-anonymization attacks are inherent. Indeed, the DP3T team answered to Vaudenay's paparazzi attack [115] in [111] claiming that "This is a known attack vector inherent to all contact tracing systems, whether centralized or decentralized (SRE, Inherent Risk 1)". Nevertheless, we disprove such claim by proposing new ACTs that are not vulnerable to the paparazzi attack. In a nutshell, the catch is that Vaudenay's adversary uses passive undetectable antennas. While infected users in DP3T can be de-anonymized even by such a weak adversary, that does not hold in general. It follows that DP3T leaks more than what is strictly required by the proximity tracing functionality.

The document [110] goes on providing a taxonomy of ACTs that were proposed up to that time. The systems are categorised w.r.t. two features: (i) whether the pseudonyms are locally generated or assigned by a central server (i.e., the ACT is centralized or decentralized), (ii) whether the infected users share their own pseudonyms or the pseudonyms that they recorded. Then, after having evaluated all the possible attacks and risks they could think of, the document proceeds in conveying the idea that the best trade-offs are offered by decentralized systems which share the pseudonyms used by the infected users. For example, they make the following consideration w.r.t. systems where "an infected person uploads all identifiers observed during the contagious window to the server". They argue that "for epochs in which groups of at least three people were in close proximity to each other, this will reveal temporal colocation information about infected individuals to the server." Although this is true for systems which submit recorded pseudonyms without any further transformation, this is in general not true. There are modifications of such approach, as we demonstrate in this thesis (cfr., Sec. 3.3.1), that avoid such leaks.

Although useful for having a general understanding of the matter, such high-level evaluations can induce in making general claims that do not have any strong foundation.

Integrity risks. Regarding false-alerts injection, the document [110] conveys the idea that all the attacks are just tied to limitations in preventing, without using location information, range extension or signal relay at the bluetooth level. However, this point of view only considers the most powerful adversary. It neglects that there could be different levels of protection w.r.t. different adversaries. In fact, a lower-level characterization could allow to better evaluate the trade-offs of different solutions.

For example, the document discusses false alarms through BLE range extensions (GR1), and false alarms through active relays (GR2). In GR1, the attacker uses her device as in the protocol specification but extends its range. Therefore, the attacker needs to find a way to be marked as infected. In GR2, the attacker simply relays (possibly back and forth) interactions with devices that are likely to be marked as infected later on. GR1 is deemed inherent because anybody could pay an infected user to upload different data. However, the risk of (much more dangerous) automated versions of this bribing attack (i.e., Vaudenay's terrorist attack) is not discussed. In Sec. 3.4, we show that such attacks are possible and can be run against practically deployed GAEN ACTs.

Mitigations to GR2 are deemed possible only via distance bounding protocols or by binding transmissions to locations. Nevertheless, there could be systems with different kinds of resilience which do not use any of these technique and just address the problem at the protocol level. For example, in Sec. 3.3, we propose ACTs that are not vulnerable to replay attacks (i.e., relay attacks in which the signal is relayed only in one direction) as opposed to DP3T/GAEN.

1.3.6 Our Contributions in ACTs

In Sec. 3.3, we investigate both known and novel privacy and integrity attacks to ACTs. In particular, since one of the main reasons advocated to prefer the decentralized approach over the centralized one was to protect citizen from mass surveillance, we take into consideration also adversaries that are able to corrupt the server and/or the health authorities. We first evaluate DP3T/GAEN w.r.t. such attacks revealing that they are vulnerable to most of them. Then, on the positive side, we propose a novel alternative ACT named Pronto-C2, as well as a lightweight version named Pronto-B2, tackling both privacy and integrity attacks. Pronto-C2 and Pronto-B2 act also as concrete examples disproving some DP3T's claims about the inherent nature of certain integrity and privacy issues in ACTs. Parts of such findings were published in [13, 14]. The prevalent contributions I gave in such co-authored publications are the formalization of attacks, threats, and security guarantees of ACTs, as well as the security analyses of DP3T, Pronto-B2, and Pronto-C2.

In Sec. 3.4, we investigate the issue of false alert injections through bribing attacks. We point out that their importance was either downplayed or ignored by major security assessments. However, we are the firsts to concretely demonstrate the viability of the terrorist attack *conjectured* by Vaudenay. In particular, we show how to overcome serious obstacles that seemed to hinder the applicability of the terrorist attack. Namely, smart contracts cannot query the contact tracing server to verify that an upload was performed, and cannot store any secret value. We propose a suite of smart contracts that implements the terrorist attack w.r.t. concretely deployed GAEN-based ACTs. We thus demonstrate that such attacks are possible and practical. Therefore, they should be taken into consideration when building ACTs. Parts of such findings were published in [16]. My prevalent contributions in such co-authored publication are the design of the attacks, as well as all the considerations on their practicality and all the involved subtleties.

1.4 Second Contribution: Threshold Cryptography

Parts of our results in the area of threshold cryptography included in this thesis were published in ESORICS 2022 [11]. In particular, such publication deals with zero-knowledge proofs over threshold relations and threshold ring signatures. The results related to ETRS will appear in the proceedings of PKC 2023. In the meanwhile, we published these results on ePrint [10].

Threshold cryptography. Threshold cryptography deals with designing cryptosystems that can be used only when an authorized group of users cooperate. A typical example is requiring the cooperation of at least a threshold of t users within a larger group of n users. Threshold primitives are crucial in applications involving multiple senders or receivers. Some examples are threshold decryption (i.e., a secret can only be unveiled by at least t receivers) [33], threshold signatures (i.e., a message can only be signed if at least t parties agree) [104], and threshold ring signatures (i.e., t parties having independent keys can sign a message while hiding their identity in a larger set of n users) [30]. With the increasing adoption of distributed systems like blockchains, these primitives are becoming even more relevant. In such scenarios, some key challenges include reducing the computational load, the size of published data, and the interaction among the parties, as well as proposing new threshold primitives with enhanced functionalities.

1.4.1 Threshold Ring Signatures

A central cryptographic primitive that can be used to provide anonymity in applications is ring signatures [98]. Ring signatures [98] are digital signatures which allow one user to sign a message while hiding her identity in a larger group called ring $\mathcal{R}$. In practice, the signing algorithm, aside the message, takes as input a set of public keys (i.e., the ring) and one of the corresponding secret keys. The produced signature guarantees that one of the public keys in the ring signed the message, while hiding which one of the secret keys was used to create the signature. Clearly, the larger is $\mathcal{R}$ the greater is the anonymity provided to the signer. A practical application of ring signature is whistleblowing. By signing a message, a member of a company can report a wrong practice of the company itself while hiding his identity among all the other employees. Constructions for ring signatures are known from a variety of cryptographic tools such as RSA [53], pairing groups [27, 44, 127], non-interactive zero-knowledge proofs [7, 28, 64], and lattices [24, 55, 85, 86].

Threshold ring signatures [30] enrich ring signatures by allowing t signers to hide their identity within the ring. The signature guarantees that t members of $\mathcal{R}$ signed the message without revealing which ones. Ring signatures can be seen as threshold ring signatures with $t = 1$.

1.4.2 Proofs over Threshold Relations

An enabler for threshold primitives are cryptographic primitives that can be exploited by a sender iff she holds at least t out of n secrets.

Then, to get a corresponding threshold primitive, one has to find a way to distribute the computation among different users, each one holding a different secret. A significant example are zero-knowledge (ZK) proofs over threshold relations.

Roughly speaking, ZK proofs are proofs that allow a prover P to convince a verifier V about the veracity of a claim without revealing nothing more. In this setting, we have an NP language $\mathcal{L}$ with corresponding poly-time relation $\mathcal{R}_{\mathcal{L}}$. P and V have as common input the statement x, moreover P gets a witness w such that $(x, w) \in \mathcal{R}_L$. The goal of P is to convince V that $x \in \mathcal{L}$, without revealing anything else. ZK proofs must also be sound, meaning that a dishonest prover cannot convince a verifier if $x \notin \mathcal{L}$. Additionally, a proof is said to be a proof of knowledge (PoK) if the prover is forced to know a witness w such that $(x, w) \in \mathcal{R}_L$ to make the verifier accept.

In proofs over threshold relations (PTRs), a statement consists of ℓ instances and the prover wants to prove knowledge of witnesses for at least k of them. For simplicity, we will refer to such a proof as a (k, ℓ)-PTR. This primitive is very useful in a number of privacy-centered applications. One example is e-voting [49], where the voter has to demonstrate that he voted for at most k out of n choices without revealing his choices. Interestingly, in certain cases a (k, ℓ)-PTR can also be used as building block for a multiple-senders primitive such as threshold ring signature. At a high level, starting from a (k, ℓ)-PTR one can get a corresponding threshold ring signature by solving two technical challenges: (i) find a way to distribute the computation of a single prover holding k witnesses to k different provers each one holding a single witness, (ii) find a way to tie a message m to the produced proof.

1.4.2.1 Σ-Protocols

In the space of interactive zero-knowledge proofs, there is a sub-class of protocols known as Σ-protocols. Σ-protocols are very popular because of their appealing features. Although there are Σ-protocols that cover all NP [25], there exists many Σ-protocols which are tied to a particular language. Such protocols trade generality for efficiency and are therefore suitable for practical applications. In the literature, there are several Σ-protocols for languages of practical relevance [47,73,89]. One of the most known is the Schnorr's protocol [101] for proving the knowledge of a discrete logarithm. Σ-protocols are 3-round public coin protocols (i.e., the verifier only sends random messages sampled from a fixed distribution). The proof of knowledge property is formulated through the notion of special soundness. It requires the existence of a poly-time extractor algorithm which, on input two protocol runs with the same first-round messages and different second-round messages, outputs a witness w s.t. $(x, w) \in \mathcal{R}_{\mathcal{L}}$. As we will see next, Σ-protocols are very versatile since, due to their structure, they can be combined together to get efficient PTRs.

1.4.3 The Chase for Compact and Practical PTRs

There has been an effort in the past to obtain PTRs for practical languages. In [48], Cramer et al. showed an efficient composition of Σ-protocols to obtain a (k, ℓ)-PTR. Their technique has several advantages. Indeed, it applies to all Σ-protocols, even for different languages, and gives again a Σ-protocol in output. However, the computation and communication costs are linear in the number of the composed protocols. Subsequently, other composition techniques have been proposed, with the aim of adding features or reducing the communication complexity. Ciampi et al. [45] added the delayed input feature, meaning that is it possible for the prover to learn the instances only just before computing the third round. Unfortunately, the resulting protocol is only computational honest verifier zero knowledge, regardless of the flavour of the composed protocols. The communication complexity is the same as [48].

Very recently, Attema et al. [7] obtained a very compact (k, ℓ)-PTR with logarithmic communication complexity. However, their result is not general since it applies only to discrete logarithms. Even more recently, Goel et al. [64] achieved a similar result in terms of communication complexity which applies to a large class of Σ-protocols (which they call stackable Σ-protocols). Nevertheless, the techniques of [64] are communication-efficient only when $k = 1$. In the full version of their paper [62], they proposed an approach for generic values of k

that does not improve the communication complexity of [48][6].

1.4.4 Extendable Threshold Ring Signatures

Some threshold ring signatures also enjoy a property called flexibility [90, 93]. They allow new signers to join already produced signatures: a signature on a message m that was already created with threshold t for a ring $\mathcal{R}$ can be transformed into a new signature on message m with threshold $t+1$ w.r.t. the same ring $\mathcal{R}$. The interesting aspect of flexible threshold ring signatures is that the update does not require the participation of any previous signer. Nevertheless, until recently, all known threshold ring signatures did not offer an analogous property that would allow extending the ring. In other words, all previous constructions required to fix the ring from the beginning and did not allow to modify it further.

This problem has been addressed for the first time in the recent work of Aranha et al. [5] which has put forth the notion of *extendable* threshold ring signatures (ETRS). ETRS, aside the join operation, also provide an *extend* operation: any signature with ring $\mathcal{R}$ can be transformed by anybody into a signature with ring $\mathcal{R}'$ s.t. $\mathcal{R} \subset \mathcal{R}'$. After the extend operation, all signers in $\mathcal{R}'$ can join the signature.

1.4.5 ETRS and Count-Me-In Applications

The new functionality offered by ETRS could in principle be exploited in anonymous count-me-in applications. In practice, a public bulletin board would be used as a synchronization medium to run an anonymous petition. An ETRS on the bulletin board - with threshold t on a message m that contains a proposition - would certify that t authentic different people out n possible people agreed with the proposition. Unlike usual petitions where the identity of the signers is known (or protected by a trusted authority), the identity of the actual t signers would be hidden among all the possible signers. This system would allow people to express their opinion on sensitive/controversial topics

[6]After our work was published in ESORICS 2022 [11], Goel et al. updated their paper [63] with a construction inspired by our work which achieves the same communication complexity.

while protecting themselves from the consequences. Furthermore, the number of supporters of a proposal would be public at all times.

The extend and join operations would then be used to update the ETRS on the bulletin board. With an extend operation anybody could augment the set of possible supporters, while with a join operation a user could anonymously support the proposition.

However, even if such applications are implicitly suggested in [5], their ETRS has a very limited notion of anonymity that does not support such applications. In their anonymity definition, the adversary is constrained in seeing only one signature that is the result of an arbitrary number of (hidden) join/extend operations. Additionally, their definitions do not guarantee anonymity w.r.t. other signers. This implies that the only secure way of using such ETRS is by requiring that *all* the possible supporters privately communicate together to produce a *unique final* ETRS which can be made public only at the very end. Furthermore, all the signers have to trust each other. All these restrictions seriously hinder the usage of ETRS in real-world count-me-in applications.

1.4.6 Our Contributions in Threshold Cryptography

In this thesis, we give several new contributions in the area of threshold cryptography. In Sec. 4.1, we focus on how to obtain compact PTRs for a large class of useful languages. We do so by devising a composition that supports a large class of Σ-protocols, namely the stackable Σ-protocols defined by Goel et al. [62]. Additionally, we show how to construct a threshold ring signature from our PTR.

In Sec. 4.2, we propose a novel ETRS with enhanced anonymity properties that are suited for real-world count-me-in applications. In the following paragraphs, we give a more detailed overview of our contributions.

Compact PTRs and Threshold Ring Signatures. In light of the state of affairs discussed in Sec. 1.4.3, we have the following open question:

Is it possible to obtain a practical (i.e., round, communication and computation-efficient) (k, ℓ)-PTR for a large class of Σ-protocols (and thus for several useful languages) with communication complexity sub-linear in ℓ preserving statistical/perfect zero knowledge?

We solve the above open problem when $k = o(\frac{\ell}{\log \ell})$ by showing how to efficiently combine the same large class of Σ-protocols considered in [62] obtaining a (k, ℓ)-PTR with communication complexity that is roughly $k \log \ell$. In scenarios where k is way smaller than ℓ (e.g., k is constant or even $\sqrt{\ell}$) this is a significant improvement. Moreover, our construction, similarly to [62], can also be used for (k, ℓ)-PTR involving Σ-protocols for different languages. The protocol obtained through our techniques is still a Σ-protocol and thus, it can be combined again with our techniques or other techniques (e.g., [48]) for composing Σ-protocols. Finally, our construction preserves the flavour of the zero-knowledge property of the composed protocols. Indeed, our (k, ℓ)-PTR called $\Pi_{\mathsf{k},\ell}$ is still statistical/perfect honest-verifier zero knowledge (HVZK) if the base Σ-protocols are statistical/perfect HVZK. Starting from our $\Pi_{\mathsf{k},\ell}$, we can get a threshold ring signature scheme. In our threshold ring signature scheme, the size of a signature corresponds roughly to $\mathcal{O}(k \log \ell)$ group elements. Nice features of our threshold ring signature include a reduced level of interaction among the signer, as well as the support for hierarchical access structures. Indeed, our PTR supports hierarchical relations that are more expressive then simple thresholds, thus achieving better anonymity properties. More details are presented in Sec. 4.1.

Parts of such results were published in [11]. My prevalent contribution in such co-authored publication was to design our (k, ℓ)-PTR and to prove its security.

ETRS with enhanced anonymity. We address the shortcomings of ETRS mentioned in Sec. 1.4.5. First, we propose stronger security definitions that guarantee anonymity even against adversaries that see the full "evolution" of a signature. Second, we propose a new ETRS construction that achieves our strong anonymity definitions, and also improves in both computational and communication complexity over previous work. Our construction relies on extendable non-interactive witness indistinguishable proof of knowledge (ENIWI PoK), a novel technical tool that we formalize and construct, and that may be of independent interest. Roughly speaking, witness indistinguishability is a weaker form of zero knowledge that only guarantees that the verifier is not able to understand which witness was used by the prover. Since threshold relations inherently have multiple witnesses, this is sufficient in several applications. Our ENIWI PoK has the interesting feature of being updatable by different provers. A prover, starting from a (k, ℓ)-PTR, can non-interactively build a (k', ℓ')-PTR with $k' \geq k$ and $\ell' \geq \ell$, using only the witnesses for the "new" $k' - k$ instances. More details are presented in Sec. 4.2.

Parts of such results will appear in the proceedings of PKC 2023. I have worked on these results during my visiting period at IMDEA Software Institute, under the supervision of Prof. Dario Fiore. My prevalent contributions in such co-authored result were to identify the open problem, formulate anonymity definitions suited for real-world applications of ETRS, define and construct the ENIWI PoK and our ETRS, and proving the security of both primitives.

Confronting the "ugly side" of provable security. When analyzing previous works we improve on, we found out some inconsistencies in the usage of the provable security paradigm. They can be described according to the issues mentioned in Sec. 1.1.

- **Problematic proofs:** We discovered a fallacy in the security reduction of the Extended Honest-Verifier Zero Knowledge property of the $(1, \ell)$-PTR of Goel et al. [62]. In particular, they presented a very high-level reduction to standard properties of commitment schemes. At a superficial examination, such proof may seem trivially correct as it resembles many similar security reductions. However, we show that the proof is incorrect and stronger properties for the commitment scheme are needed. Luckily, the concrete instantiation they proposed also fulfills our stronger definitions. Recently, Goel et al. [63] updated their paper acknowledging the issues we pointed out. This and other issues are described in detail in Sec. 4.1.6.

- **Inadequate definitions:** Regarding the ETRS proposed in [5], we did not find any technical issue. However, as we already mentioned in Sec. 1.4.5, we discovered an even more subtle problem. The paper [5] introduces ETRS as an enabler for count-me-in applications. They are supposed to allow people to easily express their opinions while staying anonymous, even from their fellow signers. However, the definitions they propose are insufficient to reach such goal. Interestingly, the issue concretely shows up in their most efficient ETRS. In such ETRS it is straightforward, looking at two subsequent signatures, to identify who joined the signature. These issues are described in detail in Sec. 4.2.1 and Sec. 4.2.3.

Chapter 2

Preliminaries

In this chapter, we introduce the notation and the basic concepts that will be used throughout the thesis. In particular, Sec. 2.1 describes the notation used in this thesis, Sec. 2.2 reports the main number-theoretic assumptions, and Sec. 2.3 reviews the main cryptographic tools used in this thesis.

2.1 Notation

We use $\mathbb{N}$ to denote the set of all natural numbers and we let PPT stand for probabilistic polynomial time. For a probabilistic algorithm A, $A(x)$ denotes the probability distribution of the output of A when run with x as input. We use $A(x;r)$ instead to denote the output of A when run on input x and randomness r. We write $A^{B(\cdot)}$ to indicate that A is given oracle access to algorithm B. This means that A can query an oracle to get the output of B on input of its choice. This is particularly useful in security games where the output of B depends also on secret information A is not given access to.

We denote with $\lambda \in \mathbb{N}$ the security parameter and with $\mathsf{poly}(\cdot)$ a positive polynomial poly. Generally, every algorithm in a cryptographic construction takes in input the security parameter 1^λ in unary. When an algorithm takes more than one input, 1^λ is omitted. We say that a function $\nu : \mathbb{N} \to \mathbb{R}$ is negligible if every positive polynomial $\mathsf{poly}(\cdot)$ and all sufficiently large λ it holds that $\nu(\lambda) \leq \frac{1}{\mathsf{poly}(\lambda)}$.

A *polynomial-time* relation $\mathcal{R}$ is a relation for which membership of (x, w) to $\mathcal{R}$ can be decided in time polynomial in $|x|$. If $(x, w) \in \mathcal{R}$ then we say that w is a *witness* for the *instance* x. A polynomial-time relation $\mathcal{R}$ is naturally associated with the NP language $\mathcal{L}_{\mathcal{R}}$ defined as $\mathcal{L}_{\mathcal{R}} = \{x | \exists w : (x, w) \in \mathcal{R}\}$. Similarly, an NP language $\mathcal{L}$ is naturally associated with a polynomial-time relation $\mathcal{R}_{\mathcal{L}}$.

A distribution ensemble $\{X(a)\}_{a \in \{0,1\}^*}$ is an infinite sequence of probability distributions, where a distribution $X(a)$ is associated with each value of a. We say that two distribution ensembles $\{X(n)\}_{n \in \mathbb{N}}$ and $\{Y(n)\}_{n \in \mathbb{N}}$ are computationally indistinguishable if for every PPT distinguisher D, there exists a negligible function ν such that for all $n \in \mathbb{N}$,

$$\Pr[\mathsf{D}(X(n), n) = 1] - \Pr[\mathsf{D}(Y(n), n) = 1] \leq \nu(n).$$

We say that $\{X(n)\}_{n \in \mathbb{N}}$ and $\{Y(n)\}_{n \in \mathbb{N}}$ are statistically indistinguishable if the above holds for all D. We use $\approx_s$ to indicate that two distributions are statistically indistinguishable. We use $\approx$ to indicate that two distributions are identically distributed.

We denote by $[n]$ for an $n \in \mathbb{N}$ the set of numbers $\{1, \ldots, n\}$. We refer to vectors using the bold font as $\mathbf{v}$. To indicate the i-th position of $\mathbf{v}$, we write $\mathbf{v}[i]$, we do the same for binary strings. When using $\leftarrow$ we mean that the variable on the left side is assigned with the output value of the algorithm on the right side. With $\leftarrow_{\$}$, we indicate that the variable on the left side is assigned a values sampled randomly according to the distribution on the right side.

When dealing with cyclic groups, we use the multiplicative notation to denote the group operation. When dealing with pairing groups, we use the additive notation to denote the group operation and the multiplicative notation to denote the pairing operation. To simplify the description, in this chapter we only use the multiplicative notation as bilinear groups are only used in Sec. 4.2. In Sec. 4.2.4, we will restate all the relevant assumptions with the additive notation in the bilinear group setting.

2.2 Number-theoretic Assumptions

In this section, we list the number-theoretic assumptions that are of interest for this thesis.

Assumption 1 (DL). *There exists a PPT algorithm* $\mathsf{GG}(1^\lambda)$ *which returns the description of a cyclic group* $\mathbb{G}$ *in which the sampling of the elements is efficient, such that for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$*:*

$$\Pr\big[\mathcal{A}(1^\lambda, \mathbb{G}, g, h) = y | \mathbb{G} \leftarrow \mathsf{GG}(1^\lambda); h \leftarrow \mathbb{G}; y \leftarrow \mathbb{Z}_{|\mathbb{G}|}; g \leftarrow h^y\big] = \nu(\lambda).$$

Let $\mathbb{G}$ be a cyclic group of order q, g generator of $\mathbb{G}$ and let A, B and X be elements of $\mathbb{G}$. 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 $0 \le \alpha, \beta \le |\mathbb{G}| - 1$ and $X = g^{\alpha\beta}$. If this is not the case, the tuple is called *non-DH.*

The *Computational Diffie-Hellman* assumption (the CDH assumption) posits the hardness of computing X given A and B.

Assumption 2 (CDH). *For every PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$*:*

$$\Pr\big[\mathcal{A}((\mathbb{G}, q, g), g^\alpha, g^\beta) = g^{\alpha\beta} | (\mathbb{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda); \alpha, \beta \leftarrow \mathbb{Z}_q\big] = \nu(\lambda).$$

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.

Assumption 3 (DDH). *For every PPT algorithm* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\Big| \Pr\big[(\mathbb{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda); \alpha, \beta, \gamma \leftarrow \mathbb{Z}_q : \mathcal{A}((\mathbb{G}, q, g), g^\alpha, g^\beta, g^\gamma) = 1\big] -$$
$$\Pr\big[(\mathbb{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda); \alpha, \beta, \gamma \leftarrow \mathbb{Z}_q : \mathcal{A}((\mathbb{G}, q, g), g^\alpha, g^\beta, g^{\alpha\beta}) = 1\big] \Big| \le \nu(\lambda).$$

2.3 Cryptographic Tools

Hash functions. A hash function Hf is a pair of PPT algorithms $(\mathsf{Gen}; \mathsf{H})$ fulfilling the following:

- Gen is a PPT algorithm which takes as input a security parameter λ and outputs a key s. We assume that 1^λ is included in s.

- There exists a polynomial poly such that H is (deterministic) polynomial-time algorithm that takes as input a key s and any string $x \in \{0,1\}^*$,and outputs a string $\mathsf{H}^s(x) \in \{0,1\}^{\mathsf{poly}(\lambda)}$.

The security property of $\mathsf{Hf} = (\mathsf{Gen}, \mathsf{H})$ is defined using an experiment $\mathsf{ExpHashColl}$ for Hf, an adversary $\mathcal{A}$ and a security parameter λ.

$\mathsf{ExpHashColl}_{\mathsf{Hf},\mathcal{A}}(\lambda)$

1. $s \leftarrow \mathsf{Gen}(1^\lambda)$.
2. $(x, x') \leftarrow \mathcal{A}(s)$
3. Output 1 if and only if $x \neq x'$ and $\mathsf{H}^s(x) = \mathsf{H}^s(x')$.

Definition 1 (Collision-resistant hash function)**.** *A hash function* $\mathsf{Hf} = (\mathsf{Gen}, \mathsf{H})$ *is collision resistant if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\Pr\big[\mathsf{ExpHashColl}_{\mathsf{Hf},\mathcal{A}}(\lambda) = 1\big] \leq \nu(\lambda).$$

Message Authentication Codes (MAC). A message authentication code (MAC) consists of a tuple of algorithms $(\mathsf{Gen}, \mathsf{Tag}, \mathsf{Verify})$ such that

- $\mathsf{Gen}(1^\lambda)$: on input the security parameter, output a key k in the key space $\mathcal{K}$.

- $\mathsf{Tag}(m, k)$: on input a message m in the message space $\mathcal{M}$ and a key k, output a tag θ.

- $\mathsf{Verify}(m, \theta, k)$: on input a message m and a key k, output 1 iff θ is a correct tag on m under key k.

A MAC must satisfy the following properties:

Definition 2 (Completeness). *A* $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Verify})$ *is complete if for every* $m \in \mathcal{M}$ *it holds that:*

$$\Pr\left[\mathsf{Verify}(m, \theta, k) = 1 \;\middle|\; \begin{array}{c} k \leftarrow \mathsf{Gen}(1^\lambda); \\ \theta \leftarrow \mathsf{Tag}(m, k); \end{array}\right] = 1.$$

Definition 3 (Unforgeability). *A MAC* $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Verify})$ *is unforgeable if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\Pr\left[\mathsf{ExpMacUnf}_{\mathsf{MAC},\mathcal{A}}(\lambda) = 1\right] \leq \nu(\lambda).$$

$\mathsf{ExpMacUnf}_{\mathsf{MAC},\mathcal{A}}(\lambda)$

1. $k \leftarrow \mathsf{Gen}(1^\lambda)$.
2. $(m, \theta) \leftarrow \mathcal{A}(1^\lambda)^{\mathsf{Tag}(\cdot,k)}$. Let $\mathcal{Q}$ be the set of all queries asked by $\mathcal{A}$ the Tag oracle.
3. Output 1 iff $m \notin \mathcal{Q}$ and $\mathsf{Verify}(m, \theta, k) = 1$. Otherwise, output 0.

Commitment schemes. A (non-interactive) commitment scheme consists of two PPT algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Gen}$: on input the security parameter, outputs public parameters pp.
- $\mathtt{com} \leftarrow \mathsf{Commit}(\mathsf{pp}, m; r)$ on input the public parameters, a message $m \in \{0,1\}^k$, and randomness $r \in \{0,1\}^\lambda$ outputs a value $\mathtt{com} \in \{0,1\}^l$.

The pair (m, r) is also called the *opening*. Intuitively, a secure commitment can be seen as a digital envelope. It satisfies two properties called binding and hiding. The first property says that it is hard to open a commitment in two different ways. The second property says that a commitment hides the underlying message.

Definition 4 (Perfect binding)**.** *We say that a commitment scheme satisfies perfect binding if for all* $\mathtt{com} \in \{0,1\}^l$ *there do not exist values* (m_0, m_1, r_0, r_1)*, with* $m_0 \neq m_1$*, s.t.* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ *and* $\mathsf{Commit}(\mathsf{pp}, m_0; r_0) = \mathsf{Commit}(\mathsf{pp}, m_1; r_1) = \mathtt{com}$.

Definition 5 (Computational hiding)**.** *We say that a commitment scheme satisfies computational hiding if for all pairs of message* $m_0, m_1 \in \{0,1\}^k$*,* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$*, it holds that*
$\{\mathsf{Commit}(\mathsf{pp}, m_0; 1^\lambda)\}_{\lambda \in \mathbb{N}} \approx_c \{\mathsf{Commit}(\mathsf{pp}, m_1; 1^\lambda)\}_{\lambda \in \mathbb{N}}$.

There are also commitment schemes with dual properties, meaning that hiding is perfect, while binding is computational. However, there cannot exist commitment schemes having both properties with a perfect flavor. There also exist interactive commitment schemes, where algorithms are replaced by protocols.

Public Key Encryption. A public key encryption (PKE) scheme is a set of PPT algorithms $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$: on input the security parameter, output a new public and secret key pair.
- $\mathsf{a} \leftarrow \mathsf{Enc}(m, \mathsf{pk})$: on input a message m, and a public key pk, output a ciphertext a.
- $m \leftarrow \mathsf{Dec}(\mathsf{a}, \mathsf{sk})$ on input a ciphertext a, and a secret key sk, output a message m.

Additionally, a public key encryption scheme is homomorphic w.r.t. a function f, if there exists a PPT algorithm that works as follows.

- $\mathsf{a}' \leftarrow \mathsf{Eval}(\mathsf{a}, x, \mathsf{pk})$: on input a ciphertext a, a message x, and the public key pk. Let $y \leftarrow \mathsf{Dec}(\mathsf{a}, \mathsf{sk})$, it returns a ciphertext a' s.t. $f(y, x) = \mathsf{Dec}(\mathsf{a}', \mathsf{sk})$.

There are several notions for the security of public key encryption schemes. One of the most used ones is indistinguishability under chosen-plaintext attacks (IND-CPA).

Definition 6 (IND-CPA)**.** *A public key encryption scheme* $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* IND-CPA *secure if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\Pr\big[\mathsf{ExpCPA}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\big] \leq \frac{1}{2} + \nu(\lambda).$$

$\mathsf{ExpCPA}_{\mathsf{PKE},\mathcal{A}}(\lambda)$

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
2. $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk})$.
3. Sample $b \leftarrow\!\!\$\ \{0,1\}$, compute $\mathsf{a} = \mathsf{Enc}(m_b, \mathsf{pk})$.
4. $b' \leftarrow \mathcal{A}(\mathsf{a})$.
5. Output 1 iff $b' = b$. Output 0 otherwise.

Digital signatures. A digital signature scheme consists of a set of algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, such that:

- $\mathsf{Gen}(1^\lambda)$: on input the security parameter, output a key pair $(\mathsf{sk}, \mathsf{pk})$, where sk is the secret key and pk the public key.
- $\mathsf{Sign}(m, \mathsf{sk})$: on input a message $m \in \{0,1\}^*$ and a signing key sk, output a signature σ.
- $\mathsf{Verify}(m, \sigma, \mathsf{vk})$: on input a message m, a signature σ, and the public key pk, output 1 if the signature σ correctly verifies under vk.

A digital signature scheme must satisfy the following properties:

Definition 7 (Completeness)**.** *A digital signature scheme* $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is complete if for every* $m \in \{0,1\}^*$ *it holds that:*

$$\Pr\left[\mathsf{Verify}(m, \sigma, \mathsf{pk}) = 1 \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda); \\ \sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}); \end{array}\right] = 1.$$

Definition 8 (Unforgeability)**.** *A digital signature scheme* $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is unforgeable if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\Pr\left[\mathsf{ExpDsUnf}_{\mathsf{DS},\mathcal{A}}(\lambda) = 1\right] \leq \nu(\lambda).$$

$\mathsf{ExpDsUnf}_{\mathsf{DS},\mathcal{A}}(\lambda)$

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. $(m, \sigma) \leftarrow \mathcal{A}(1^\lambda)^{\mathsf{Sign}(\cdot, \mathsf{sk})}$. Let $\mathcal{Q}$ be the set of all queries asked by $\mathcal{A}$ the Sign oracle.
3. Output 1 iff $m \notin \mathcal{Q}$ and $\mathsf{Verify}(m, \sigma, \mathsf{pk}) = 1$. Otherwise, output 0.

2.3.1 Proof Systems

We present here a high-level overview on proof system- We take many definitions from [119], to which we refer the reader for more details. We define the notion of proof system for a language $\mathcal{L}$. There are two parties, namely prover P and verifier V. Both P and V are PPT interactive algorithms, the language $\mathcal{L}$ is in NP, P and V have common input x, moreover P knows a witness w such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$. We

Definition 9 (Proof System)**.** *A proof system* $\Pi = (\mathsf{P}, \mathsf{V})$ *for an NP-language* $\mathcal{L}$ *is a pair of PPT interactive algorithms satisfying the following properties.*

Completeness: *for all* $x \in \mathcal{R}_{\mathcal{L}}$ $\Pr[\langle \mathsf{P}(w), \mathsf{V}\rangle(x) = 1] = 1.$

Soundness: *there exists a negligible function ν such that for every $x \notin \mathcal{L}$ and for every adversary P^** $\Pr[\langle \mathsf{P}^*, \mathsf{V}\rangle(x) = 1] = 0$.

Definition 10 (Witness Indistinguishability)**.** *A proof system $\Pi = (\mathsf{P}, \mathsf{V})$ is Witness Indistinguishable (WI) for a relation $\mathcal{R}_{\mathcal{L}}$, if for every malicious verifier V^*, there exists a negligible function ν such that for all x, w, w' such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$ and $(x, w') \in \mathcal{R}_{\mathcal{L}}$*

$$\left|\Pr\left[\left\langle \mathsf{P}(w, 1^\lambda), \mathsf{V}^* \right\rangle(x) = 1\right] - \Pr\left[\left\langle \mathsf{P}(w', 1^\lambda), \mathsf{V}^* \right\rangle(x) = 1\right]\right| < \nu(\lambda)$$

Definition 11 (Zero Knowledge)**.** *A proof system $\Pi = (\mathsf{P}, \mathsf{V})$ for an NP-language $\mathcal{L}$ is Zero Knowledge (ZK) if there exists an expected PPT algorithm $\mathcal{S}$ such that for any PPT algorithm V^* , any $(x, w) \in \mathcal{R}_{\mathcal{L}}$ and any $z \in \{0, 1\}^*$*

$$\{\langle \mathsf{P}(w), \mathsf{V}^* \rangle(x)\} \approx_s \{\mathcal{S}^{\mathsf{V}^*}(x, z)\}.$$

More on soundness. The soundness notion formulated above considers very powerful malicious provers having unbounded computation capabilities. A relaxed soundness notion that is very useful in practice focuses on P^* running in polynomial time. Proof systems for which the soundness requirement only holds computationally are called *argument* systems. In the remainder of this thesis, we will use the term proof also to denote argument systems and make the distinction explicit whenever is useful for the discussion. It is important to notice that the soundness requirement only considers $x \notin \mathcal{L}$. There are situations in which a claim can never be false and thus the soundness property alone is not effective. For this reason (and others), it may be crucial not only that P convinces V, but also that any P^* that convinces V for some $x \in \mathcal{L}$ must know a witness w such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$. Such notion, called proof (or argument) of knowledge, is usually formulated by the mean of a poly-time extraction algorithm. Σ-Protocols are an example of proofs of knowledge (cfr., Section 2.3.1.1).

More on zero knowledge. The flavour of zero knowledge described above is called statistical zero knowledge. If the two distributions are only computationally indistinguishable, then one obtains a different flavour called computational zero knowledge. The same applies to witness indistinguishability. Additionally, $\mathcal{S}^{\mathsf{V}^*}$ means that the simulator has only black-box access to the verifier, while z represents an auxiliary input to the verifier. Adding an auxiliary input to V^* is crucial to ensure that zero knowledge is preserved even if several executions of the protocol are performed one after another [65].

Interactive zero-knowledge proofs are deniable. The zero knowledge notion given above requires the existence of a simulator algorithm that creates protocol transcripts that are indistinguishable to regular interactions with an honest prover. This means that actual conversations V^* had with P are useless in proving to a third party that such interactions actually happened. Indeed, V^* could have generated such transcripts by itself running the simulator. This also ensures that V^* is not able to re-use such conversations to convince a thid party that $x \in \mathcal{L}$.

Non-Interactive Zero Knowledge (NIZK). As opposed to what said in the previous paragraph, there can be cases where it is essential that the prover is able to compute a proof that is ZK but at the same time is verifiable by many different verifiers. There exist systems that solve this problem considering a scenario where a single message is generated by the prover. Such message constitutes a proof that can be verified many times by many independent verifiers. Such proofs are known as Non-Interactive Zero-Knowledge (NIZK) proofs. NIZK proofs can exist in the plain model only for trivial languages[1]. There are NIZK proofs in the CRS model [26, 58, 71, 72]. There also exist NIZK arguments in the ROM as shown in the next section.

[1]Being the proof made of only one message, the malicious prover could use a simulator that outputs accepting proofs for any statement to violate soundness. If the PPT simulator outputs accepting proofs only for $x \in \mathcal{L}$, it can be used to efficiently decide membership in $\mathcal{L}$, thus $\mathcal{L}$ is in BPP.

2.3.1.1 Σ-Protocols

We consider a *3-round* public-coin protocol Π for an NP language $\mathcal{L}$ with a poly-time relation $\mathcal{R}_{\mathcal{L}}$. $\Pi = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$ is run by a prover running auxiliary algorithms $\mathsf{P}_0, \mathsf{P}_1$ and a verifier running an auxiliary algorithm V. The prover and the verifier receive common input x and the security parameter. The prover receives as an additional private input a witness w for x. Prover and verifier use the auxiliary algorithms $\mathsf{P}_0, \mathsf{P}_1, \mathsf{V}$ in the following way:

1. The prover runs P_0 on common input x, private input w, randomness R, and outputs a message a. The prover sends a to the verifier;

2. The verifier samples a random challenge $c \leftarrow_{\$} \{0,1\}^{\lambda}$ and sends c to the prover;

3. The prover runs P_1 on common input x, private input w, first-round message a, randomness R, and challenge c, and outputs the third-round message z, which is then sent to the verifier;

4. The verifier outputs 1 if $\mathsf{V}(x, a, c, z) = 1$, and rejects otherwise.

The transcript (a, c, z) for the protocol $\Pi = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$, and common statement x is called *accepting* if $\mathsf{V}(x, a, c, z) = 1$.

Definition 12 (Σ-protocol). *A 3-round public-coin protocol* $\Pi = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$, *is a Σ-protocol for an NP language* $\mathcal{L}$ *with a poly-time relation* $\mathcal{R}_{\mathcal{L}}$ *iff the following properties are satisfied*

Completeness: *For all* $x \in \mathcal{L}$ *and* w *such that* $(x, w) \in \mathcal{R}_{\mathcal{L}}$ *it holds that:*

$$\Pr\left[\mathsf{V}(x,a,c,z) = 1 \;\middle|\; \begin{array}{c} R \leftarrow_{\$} \{0,1\}^{\lambda}; c \leftarrow_{\$} \{0,1\}^{\lambda}; \\ a \leftarrow \mathsf{P}_0(x, w; R); \\ z \leftarrow \mathsf{P}_1(x, w, a, c; R) \end{array} \right] = 1.$$

Special Soundness: $\exists$ *PPT* $\mathsf{Extract}$, *such that on input* x *and two accepting transcripts* (a, c_0, z_0) *and* (a, c_1, z_1) *for* x, *where* $c_0 \neq c_1$, *it holds that*

$$\Pr[(x, w) \in \mathcal{R}_{\mathcal{L}} | w \leftarrow \mathsf{Extract}(x, a, c_0, c_1, z_0, z_1)] = 1.$$

Special Honest-Verifier Zero-Knowledge (SHVZK): *There exists a PPT simulator $\mathcal{S}$ that, on input an instance $x \in \mathcal{L}$ and challenge c, outputs (a, z) such that (a, c, z) is an accepting transcript for x. Moreover, the distribution of the output of $\mathcal{S}$ on input (x, c) is computationally/statistically/perfectly indistinguishable from the distribution obtained when the verifier sends c as challenge and the prover runs on common input x and any private input w such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$.*

All Σ-Protocols are WI. In [48], it is proved that SHVZK implies WI. Therefore, every Σ-Protocols is WI. We remark that while SHVZK is formulated w.r.t. a honest verifier, in WI the verifier can be malicious.

Definition 13 (Computational Σ-protocol). *A 3-round public-coin protocol $\Pi = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$ is a computational Σ-protocol for an NP language $\mathcal{L}$ with a poly-time relation $\mathcal{R}_{\mathcal{L}}$ if and only if it is complete, (computational/statistical/perfect) special honest-verifier zero knowledge, and computational special sound. Computational special soundness is specified below.*

Computational Special Soundness: $\exists$ *PPT* $\mathsf{Extract}$ *s.t.* $\forall$ *PPT* P^* $\exists$ *a negligible function $\nu(\cdot)$ such that $\forall x \in \mathcal{L}$ it holds that*

$$\Pr\big[\mathsf{ExpExt}_{\mathsf{P}^*,\mathsf{Extract}}(x) = 1\big] \leq \nu(|x|).$$

$\mathsf{ExpExt}_{\mathsf{P}^*,\mathsf{Extract}}(x)$

1. $(a, c_0, c_1, z_0, z_1) \leftarrow \mathsf{P}^*(x)$.
2. *If* $c_0 \neq c_1$, *or* $\mathsf{V}(x, a, c_0, z_0) = 0$, *or* $\mathsf{V}(x, a, c_1, z_1) = 0$ *return 0.*
3. $w \leftarrow \mathsf{Extract}(x, a, c_0, c_1, z_0, z_1)$.
4. *Return* 1 *if* $(x, w) \notin \mathcal{R}_{\mathcal{L}}$. *Otherwise, return 0.*

From now on, we will refer both to Σ-protocols and computational Σ-protocols simply as Σ-protocols. We will instead clearly state the considered flavour whenever it is useful for the discussion.

Fiat-Shamir transform. Fiat and Shamir [60] define a transformation, commonly known as the Fiat-Shamir transform, which converts every public-coin protocol such as Σ-protocols into NIZK arguments in the random oracle model. The idea is really straightforward. Since in a Σ-protocol the verifier is trusted to provide a uniformly random challenge, one can replace it with a random oracle whose output is indeed a uniformly random string. The prover would give in input to the random oracle the statement and the first message of the Σ-protocol, and use its output as the verifier's challenge. The resulting protocol requires no interaction, P sends (a, z) to V who computes c using the random oracle and runs $\mathsf{V}(x, a, c, z)$ as usual. Regarding soundness, the only difference with the interactive version is that P can locally query the random oracle as many time as he wants. Therefore, soundness now only holds computationally but if the challenge set is exponentially large and P runs in polynomial time, P would not be able to cheat. Regarding zero-knowledge, it is straightforward to observe that the random oracle removes any influence of a cheating verifier: Σ-protocols are zero-knowledge if the challenge is uniformly chosen, and this is exactly what the random oracle does. This is an example of how a Σ-protocol which is only HVZK can be used to construct a zero-knowledge proof system secure w.r.t. a malicious verifier.

2.3.1.2 Non-interactive Witness Indistinguishable Proof of Knowledge

We now define the notion of Non-Interactive Witness Indistinguishable (NIWI) Proof of Knowledge (PoK) in the CRS model. Let us consider an NP language $\mathcal{L}$ with associated poly-time relation $\mathcal{R}_{\mathcal{L}}$. A non-interactive proof system consists of the following algorithms:

- $\mathsf{gk} \leftarrow\$ \mathsf{Gen}(1^\lambda)$: on input the security parameter, output a group key.

- $\mathsf{crs} \leftarrow \mathsf{CRSSetup}(\mathsf{gk})$: on input the group key, output a common reference string $\mathsf{crs} \in \{0, 1\}^\lambda$.

- $\Pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$: on input statement x and witness w s.t. $(x, w) \in \mathcal{R}_{\mathcal{L}}$, output a proof Π.

- $0/1 \leftarrow \mathsf{PrVerify}(\mathsf{crs}, x, \Pi)$: on input statement x and proof Π, output either 1 to accept or 0 to reject.

- $\Pi' \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \Pi)$: on input statement x and proof Π for $x \in \mathcal{L}$, output a randomized proof Π'.

A non-interactive proof system is said to be a witness indistinguishable (NIWI) if all the properties below are satisfied.

Definition 14 (Completeness)**.** *A proof system for* $\mathcal{R}_\mathcal{L}$ *is complete if* $\forall \lambda \in \mathbb{N}$, $\mathsf{gk} \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$, $\mathsf{crs} \leftarrow \mathsf{CRSSetup}(\mathsf{gk})$, $(x, w) \in \mathcal{R}_\mathcal{L}$, *and* $\Pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$ *it holds that*

$$\Pr[\mathsf{PrVerify}(\mathsf{crs}, x, \Pi) = 1] = 1$$

Definition 15 (Witness Indistinguishability)**.** *We say that the proof system is witness indistinguishable (WI), if the following holds. For all* (x, w_1, w_2) *such that* $(x, w_1), (x, w_2) \in \mathcal{R}_\mathcal{L}$, *the tuples* (crs, Π_1) *and* (crs, Π_2), *where* $\mathsf{crs} \leftarrow \mathsf{CRSSetup}(\mathsf{gk})$, $\mathsf{gk} \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ *and for* $i \in [2]$, $\Pi_i \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w_i)$, *are indistinguishable. If the two tuples are identically distributed, we say that the proof system is perfect WI.*

Definition 16 (Soundness)**.** *For all PPT* $\mathcal{A}$, *and for* $\mathsf{crs} \leftarrow \mathsf{CRSSetup}(\mathsf{gk})$, $\mathsf{gk} \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$, *the probability that* $\mathcal{A}(\mathsf{crs})$ *outputs* (x, Π) *such that* $x \notin \mathcal{L}$ *but* $\mathsf{PrVerify}(\mathsf{crs}, x, \Pi) = 1$, *is negligible.*

Additionally, a NIWI is said to be a NIWI proof of knowledge (PoK) if the adaptive extractable soundness defined below is also satisfied.

Definition 17 (Adaptive Extractable Soundness)**.** *There exists a polynomial-time extractor* $\mathsf{Ext} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$ *with the following properties:*

- $\mathsf{Ext}_1(\mathsf{gk})$ *outputs* $(\mathsf{crs}_\mathsf{Ext}, xk)$ *such that* $\mathsf{crs}_\mathsf{Ext}$ *is indistinguishable from* crs *obtained running* $\mathsf{crs} \leftarrow \mathsf{CRSSetup}(\mathsf{gk})$.

- *For all PPT* $\mathcal{A}$, *the probability that* $\mathcal{A}(\mathsf{crs}_\mathsf{Ext}, xk)$ *outputs* (x, Π) *such that* $\mathsf{PrVerify}(\mathsf{crs}, x, \Pi) = 1$ *and* $(x, w) \notin \mathcal{R}_\mathcal{L}$ *where* $w \leftarrow \mathsf{Ext}_2(\mathsf{crs}_\mathsf{Ext}, xk, x, \Pi)$ *is negligible.*

Additionally, a NIWI is also randomizable if it satisfies the below property.

Definition 18 (Re-randomizable Proof System)**.** *Consider the following experiment:*

- $\mathsf{gk} \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$
- $\mathsf{crs} \leftarrow \mathsf{CRSSetup}(\mathsf{gk})$
- $(x, w, \Pi) \leftarrow \mathcal{A}(\mathsf{crs})$
- *If either* $\mathsf{PrVerify}(\mathsf{crs}, x, \Pi) = 0$ *or* $(x, w) \notin \mathcal{R}_{\mathcal{L}}$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\!\!\$\ \{0, 1\}$*.*
  - *If* $b = 0$ $\Pi' \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$*.*
  - *If* $b = 1$ $\Pi' \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \Pi)$*.*
- $b' \leftarrow \mathcal{A}(\Pi')$

We say that the proof system is re-randomizable if for every PPT $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$, such that $\Pr[b = b'] \leq 1/2 + \nu(\lambda)$*.*

Chapter 3

Automatic Contact Tracing

In this chapter, we consider the recent development of automatic contact tracing systems (ACTs). In particular, we focus on DP3T/GAEN (cfr., Sec. 3.1), the most deployed ACT, and elucidate its privacy and integrity weaknesses, with a focus on mass surveillance attacks (cfr., Sec. 3.3.3 and Sec. 3.3.4). Then, we propose new ACTs, named Pronto-C2 and Pronto-B2, with improved privacy and integrity guarantees (cfr., Sec. 3.3.5). Our ACTs also act as concrete examples disproving some claims the DP3T team made [110] in their informal analysis about the security of ACTs (cfr., Sec. 3.3.8). Subsequent analyses of ACTs in the provable security paradigm [34] have also confirmed our results (cfr., Sec. 3.3.9).

Finally, we focus on a false alert injection attack called the terrorist attack which was conjectured by Vaudenay [116] (cfr., Sec. 3.4). We show how to implement such attack, with the help of smart contracts, w.r.t. concretely deployed GAEN-based ACTs. Our smart contracts implement automated black markets in which terrorists can bribe infected users to upload data that would cause false alerts in the ACT. This trade can occur without the need of the parties to know each other. The smart contracts guarantee no party can be cheated, and that there is basically no risk of being caught by the authorities. By leveraging come peculiarities of GAEN, we bypass the inherent limitations of smart contracts of not being able to query web services (cfr., Sec. 3.4.2). Exploiting decentralized oracles [126], we also show that trivial modifications of GAEN do not prevent the attack (cfr., Sec. 3.4.3). Our work demonstrates that this neglected attack is a concrete risk for ACTs as it could concretely jeopardize their integrity.

Parts of the results presented in Sec. 3.3 were published in [13, 14], while the results of Sec. 3.4 were published in [16].

3.1 Brief Description of DP3T

In this section, we briefly overview the DP3T's ACTs as reported in their white paper [114]. We describe two versions of the system: the first one, termed as "low-cost", is more efficient but provides less privacy guarantees than the second one which is termed "unlinkable". There was also another design proposed by the DP3T team. It requires to secret share the pseudonyms and to transmit the shares across multiple BLE advertisements. However, the DP3T team itself subsequently dismissed this design principle as not practical due to issues that occur at the BLE layer [109].

Low-cost design. Smartphones broadcast locally generated ephemeral pseudonyms (EphIDs) via BLE advertisements. Whenever a smartphone detects an incoming EphID, it locally stores this pseudonym EphID along with a coarse time information and every data which might be needed later to compute the risk of contagion (e.g., signal strength, duration of the contact). As the word ephemeral suggests, the pseudonyms are periodically changed to prevent tracking.

All the EphIDs that a device will ever generate can be deterministically derived from a short uniformly random secret key sk_0. At each day t, a new secret key is derived as $\mathsf{sk}_t = H(\mathsf{sk}_{t-1})$, where H is a cryptographic hash function.

Starting from sk_t, the whole set of EphIDs for day t is determined partitioning in 16-byte chunks a string whose length depends on how frequently the EphIDs are changed. Such string is computed as $\mathsf{PRG}(\mathsf{PRF}(\mathsf{sk}_t, c))$ where PRF is a pseudo-random function, c is a fixed public string, and PRG is a stream cipher. The EphIDs obtained with this procedure will be eventually broadcast in random order.

When a user is tested positive, she uploads the pair (sk_t, t) to a backend server which is trusted to provide this information to all other users and to check that the uploads are performed by authorized users, preventing the dissemination of false alerts. In [112], three candidate authorization mechanisms are proposed. After this step, the infected user's device generates a new random secret key sk_0.

Each device can periodically query the backend server in order to get the new pairs that have been added to the system. Given these pairs, the device can generate the corresponding values EphIDs seeking for matches in its local contact database. If a match is found, the risk of infection is computed given the auxiliary information and the user is notified when needed. GAEN basically follows the same design. The most remarkable difference is that instead of having a single key for the entire contagious time window, GAEN uses a fresh key for each day. Suck keys are called Temporary Exposure Keys (TEKs).

Unlinkable design. In the unlinkable design, better privacy guarantees are traded for a larger volume of downloads and storage space needed by the smartphone.

In this design, the EphIDs are randomly and independently generated in the following manner: when a new ephemeral pseudonym is needed, the smartphone generates the ephemeral pseudonym EphID_i as $\mathsf{TRUNCATE}_{128}(H(\mathsf{seed}_i))$.

Smartphones store all the seeds used in the contagious time window. When a patient is tested positive, she can selectively decide which pseudonyms she wants to communicate to the server (e.g., she can exclude pseudonyms used in the presence of a specific person).

After this decision has been made, the smartphone uploads the set composed by the selected pairs (seed_i, i). Upon receiving them, the server computes $H(\mathsf{TRUNCATE}_{128}(H(\mathsf{seed}_i))||i)$ for each pair and inserts it in a Cuckoo filter[1]. Such filters are generated and made available to the users on a regular basis.

Each smartphone uses these filters to determine whether contacts with infected individuals occurred. To do so, the smartphone checks the inclusion into the filters of its recorded ephemeral pseudonyms.

[1]A Cuckoo filter is a space-efficient probabilistic data structure used to test whether an element is a member of a set. False positive matches are possible, but false negatives are not.

3.2 Related Work

In this thesis, we mainly focus on the security of the systems proposed by the DP3T team [114]. However, the attacks we present are significant to other decentralized ACTs such as MIT-PACT [94], UW-PACT [37] and TCN [108]. Since the first proposals of automatic contact tracing systems came out, researchers started to investigate their security. Many of these works focused on the analysis of DP3T [114]. Due to its great similarity with GAEN, which is almost identical to the “low-cost design” of DP3T, practically all the results about it also apply to GAEN-based ACTs. We review privacy and integrity attacks to DP3T, and we also mention alternative ACTs that were proposed in the literature.

Privacy attacks to DP3T/GAEN. Several vulnerabilities of the DP3T systems have been analysed in various works. Tang [107] observes that the DP3T systems may be subject to identification attacks and presents a comprehensive survey on ACTs. In [115], Vaudenay presents both privacy and integrity attacks. Regarding privacy attacks, the most concerning one is the paparazzi attack. In his work, Vaudenay presented the paparazzi attack as a deanonymization attack, however it can also be seen as a tracking attack. Basically, in DP3T it is very easy for an attacker to track the locations visited by the infected users during the contagious time window. The attack has two phases. In the first phase, the attacker places passive devices[2] recording the pseudonyms broadcast over the BLE channel over a territory of his interest. In the second phase, all of these pseudonyms are matched with the ones derived from the keys distributed by the official contact tracing server. Since DP3T basically announces all the pseudonyms that an infected user used during the contagious time window, the attacker can correlate the pseudonyms used by an individual with the location he visited in that time span. GAEN mitigates this problem by reducing the time window of the tracking to one day (i.e., instead of uploading one key for the entire contagious time win-

[2]A passive device is an antenna that works only in reception mode and does not broadcast any signal. Additionally, it is able to store the received pseudonyms, along with a time stamp.

dow, infected users upload one key per day). However, this assumes that the keys are properly mixed by the server. Additionally, if the attacker colludes with the server (i.e., the adversary is the government itself) this mitigation is worthless. Seiskari shows a proof-of-concept implementation of the paparazzi attack [102]. The GitHub repository includes code to replicate the experiment. Baumgärtner et al. [19] also provide empirical evidence for the paparazzi attack in GAEN.

Another key privacy threat of ACTs is the potential disclosure of encounters, also known as the social graph. In practice, one does not want the system to reveal that two infected users met each other, or that two infected users met the same individual. Although in general decentralized systems reduce the portion of the social graph that may be disclosed, they lack of any form plausible deniability. In particular, [96] shows that users of DP3T system can easily provide digital evidence of contacts with infected users. It is true that this is only one edge of the graph, but it comes together with a proof that can be potentially used to blackmail infected users.

Integrity attacks to DP3T/GAEN. In the context of ACTs, preserving integrity basically means preventing false alerts. DP3T/GAEN systems are vulnerable to replay attacks [67,96,115]. In this case, the pseudonyms sent by an individual considered at risk (e.g., a person who is taking a test) are transmitted by an adversary to a different location in order to create fake proximity contacts. GAEN has a pretty large time window (about 2 hours [67]) for pseudonyms to be replayed successfully[3]. Baumgärtner et al. [19] provide empirical evidence of the concrete feasibility of replay attacks in GAEN. The attack can have a specific target but can also be performed at large scale. Pietrzak et al. [9] analyze inverse-sibyl attacks in which multiple adversaries cooperate to use the same pseudonyms. If one of the attackers gets to upload his data, many false alerts may be raised. This attack could be used in combination with the replay attack to increase the number of affected targets.

[3]In DP3T such time window would be one day long. In a nutshell, GAEN has a slightly better resilience to replay attacks because the pseudonyms derived from a key are broadcast in fixed order instead of a random one. See [67] for more details.

There are other attacks that can inject false positives into an ACTs without requiring to be in close proximity with likely infected users. Iovino et al. [80] concretely demonstrate the possibility to inject false alerts by replaying released TEKs. In particular, pseudonyms associated with already published TEKs are transmitted to smartphones whose clock is corrupted in order to make them believe that these pseudonyms are still valid for risk matching. The attack by Iovino et al. [80] is only possible if the ACT is vulnerable to replay attacks.

Finally, another class of attacks leading to false alerts involves bribing. Vaudenay envisions various possibilities for the development of dark economies [116] which could support false alert injection attacks, allowing them to be carried out at very large scales. In particular, the *lazy student* attack is connected to replay attacks. It is based on a dark economy where a hunter (i.e., seller) collects pseudonyms of individuals who will likely become infected later on, and deposits them on a smart contract. If the TEKs corresponding to such pseudonyms are uploaded to the server of the contact tracing system, the hunter gets a reward paid by a buyer (i.e., the lazy student). If replay attacks are doable, the buyer can use them to make target victims' apps raise false alerts. This dark economy is sustainable only if the smart contract has a way to check that pseudonyms were actually reported to the official server. Another form of dark economy described by Vaudenay is the *terrorist* attack. It involves users reporting pseudonyms that differ from the ones used during the previous days. In fact, in GAEN there is no mechanism forcing users to upload genuine TEKs. Again, a TEK could be posted on a smart contract automatically issuing a reward to whoever reports it to the contact tracing system. This purchase may lead to a massive amount of fake notifications, without relying on replay attacks.

On the (missing) risk assessment of the terrorist attack. The impact of false injection attacks seems to have gone unnoticed or just ignored. In [82] the cybersecurity risks of contact tracing systems are reviewed and compared using a subjective scoring scheme. The report considers injection of false alerts notifications by only mentioning replay attacks or trivial attacks such as recruiting people with symptoms, while the terrorist attack is not even mentioned.

Vaudenay and Vuagnoux expressed these and other concerns in their analysis of SwissCovid (i.e. the Swiss GAEN-based ACT) [117, 118]. The Swiss National Cyber Security Center (NCSC) answered to their criticism seemingly downplaying those risks. The possible development of dark economies was ignored[4], and a recap table on security issues reports on SwissCovid marks the concerns expressed by Vaudenay as addressed, including false alert injection attacks[5]. Nevertheless, no solution or mitigation to such problems is reported.

Alternatives to DP3T/GAEN. Many researchers have proposed alternative ACTs with the aim of either solving or mitigating the above issues. Pietrzak [96] and Vaundenay [115] proposed solutions and mitigations to replay and relay attacks against DP3T. We proposed Pronto-C2, and a lightweight version named Pronto-B2, to tackle both privacy and integrity attacks [12] (cfr., Sec. 3.3.5). Other systems that have some similarities to Pronto-C2 are WeTrace [35] and TraceCORONA [106]. Notably, INRIA (the French national research centre of computer science and automatics) proposed DESIRE [79], a hybrid system between the centralized and the decentralized architecture which takes many design choices from Pronto-C2. Differently from Pronto-C2, the users' at-risk status is computed on the central server instead of users' smartphones. Unfortunately, due the lack of flexibility of GAEN no country could have incorporated any of the proposed improvements in their own ACTs.

[4]Swiss National Cyber Security Center: Security Issue Submission [INR-4434]. Detailed analysis. https://www.melani.admin.ch/dam/melani/de/dokumente/2020/INR-4434_NCSC_Risk_assessment.pdf.download.pdf/INR-4434_NCSC_Risk_assessment.pdf. Accessed in June 2021.

[5]Swiss National Cyber Security Center: SwissCovid Proximity Tracing System - Public Security Test, page 8. https://www.melani.admin.ch/dam/melani/de/dokumente/2020/SwissCovid_Public_Security_Test_Current_Findings.pdf.download.pdf/SwissCovid_Public_Security_Test_Current_Findings.pdf. Accessed in June 2021.

3.3 Towards Secure ACTs: Pronto-C2 and Pronto-B2

Starting from the inspiring list of attacks presented by Vaudenay [115], we first analyze the degree of privacy and integrity protection achieved by the DP3T systems. In some of the attacks a government through its natural power controls (even partially) the server, the laboratories that detect infections, or the national territory to realize mass surveillance programs.

In the DP3T systems, one can be traced even when walking alone, without having any clue that there is somebody tracking her movements. We call this silent tracking. Indeed, a passive antenna can detect pseudonyms without transmitting anything, and can later on check if a sniffed pseudonym belongs to the list of infected persons. It is easy to link the real identity of an infected person to the pseudonyms she used in the contagious time window. Indeed, such antennas can also be installed nearby places where the citizen has to identify himself (e.g., by showing an ID card or paying with a credit card) and this allows to connect pseudonyms to identities. We believe that this is an open door to help mass surveillance programs. Additionally, BLE devices that are already in use for other purposes (e.g., information kiosks) may be used to track people. Obviously, one can not expect that nothing else will be done with BLE except contact tracing, and thus preserving privacy while other uses of BLE continue is a necessary goal. Furthermore, the use of active kiosks running precisely the BLE-based contact tracing protocol is actually recommended in [94]. Instead, we believe that they should not be put in place. Indeed, misbehaving kiosks would be indistinguishable from regular users and thus the notifications sent to these "simulated users" could be potentially used to hinder citizens' privacy.

Next, we present Pronto-B2 and Pronto-C2, two novel decentralized privacy-preserving ACTs based on BLE. We show that our systems are arguably more resilient than the DP3T systems against both mass surveillance attacks and integrity attacks. Furthermore, our ACTs practically disprove several claims about the security of ACTs made by the DP3T team [110].

3.3.1 High-Level Overview of Pronto-B2 and Pronto-C2

Our main idea can be seen as a paradigm shift compared to the approach of DP3T. Indeed, instead of asking infected people to hand over their keys to the Big Brother, we allow citizens to anonymously and confidentially call each other in the presence of the Big Brother. The way we do it is explained below. We first discuss Pronto-C2 and then describe how we can get a more practical system that we call Pronto-B2 by relaxing some privacy guarantees, still outperforming DP3T systems in terms of privacy and integrity guarantees.

3.3.1.1 Pronto-C2

In the 70s Merkle, Diffie and Hellman invented public-key cryptography. Starting with Merkle's puzzles, Diffie and Hellman proposed a key exchange protocol [51] (i.e., the Diffie-Hellman protocol) where two parties can establish a secret key K by just sending one message each on a public channel. A message consists of a group element in a setting where the Decisional Diffie-Hellman assumption holds.

In our view, the most natural way to realize a privacy-preserving ACT system consists of having as pseudonym a group element that corresponds to a message in the DH protocol. This natural idea was also proposed to the DP3T team by the GitHub user a8x9[6]. In order to actually realize such form of ACT system, one needs to solve the following two main problems.

Anonymous call: realizing a mechanism that allows an infected party to use K in order to call the other party in a secure and privacy-preserving way.

Shortening pseudonyms: making sure that the size of a group element fits the number of available bits in a BLE identifier beacon.

Calling (anonymously) the infected person. We solve the first problem by asking the infected party, after having received a proper

[6]The GitHub issue opened by a8x9 can be found at https://github.com/DP-3T/documents/issues/66

authorization from the laboratory that detected the infection, to upload K along with the authorization to a bulletin board. The bulletin board can be just managed by a server as in the DP3T systems, but ideally it should be implemented through a blockchain so that we can decentralize the server, making the entire process more transparent and reliable [7]. We suggest the use of digital signatures to implement the upload authorization mechanism. In order to make the upload of K unlinkable with the real identity of the infected person, we suggest the use of blind signatures [41]. The basic idea is that laboratories receive from the government some unpredictable activation codes that are then one by one given to infected persons. Then, an infected person connects to a service in order to exchange the authorization code with some blind signatures that will be then used to upload on the bulletin board data associated to calls. Notice that the approach of Pronto-C2 is completely different from the one adopted in the DP3T systems. Indeed, while in the DP3T systems the pseudonyms of the infected person are broadcast to everyone we instead ask the infected party to send a message that is understandable uniquely by the party she was in close proximity to. Therefore, K is like a phone call through which the infected party sends to the answering party the following message[8]: "Hello, it is you that were next to me...and I've just discovered that I'm infected".

Every person that is not infected will connect to the server and will download the recently uploaded keys to search for K (data don't need to be stored, the search can happen while downloading data). Notice that there is roughly a different key K to check for every BLE identifier beacon received in the contagious time window. This step should be preferably performed while the phone is connected to a Wi-Fi network. Moreover, for those cases where the daily amount of data to download is excessive, one can think of specifying target states/regions in the country, in order to manage a restricted amount of information. In this case, a call would also specify a corresponding state/region.

[7]In this work, when referring generically to a blockchain we always mean a permissioned blockchain (e.g., Hyperledger Fabric [3]).

[8]The Italian word "Pronto" stays for "Hello" and C2 pronounced in English sounds like "it is you" in Neapolitan language, as in the title of a very popular song by Nino D'Angelo https://www.youtube.com/watch?v=8DP3UyDS0Ts.

We remark that avoiding that two smartphones with pseudonyms A and B upload the same K is straightforward: A can just upload $H(K||A||B)$ while B can just upload $H(K||B||A)$, where H is a cryptographic hash function.

Shortening pseudonyms. Current standards suggest at least 256 bits for a group element to safely run the DH protocol over elliptic curves. However, this size exceeds the space available in a BLE identifier beacon. One might think to solve the issue of the small space in a BLE identifier beacon by just resorting to very short (and therefore in our view too risky in case of mass surveillance attacks) keys or by splitting the information into multiple identifier beacons that rotate quickly. Obviously Pronto-C2 can work smoothly with such workarounds, but since they all bring some issues, we propose a different approach that allows to use as many bits as desired for the group element, while still using only a single standard BLE identifier beacon.

In Pronto-C2, we decouple the group element from the pseudonym precisely like in operating systems a large amount of data is represented by a pointer. Recall that a value announced in a BLE identifier beacon should last only for a few minutes, to then be replaced by a new one. The smartphone will periodically generate new independent group elements for DH and will keep them locally. Since such group elements are too large to be sent in BLE identifier beacons, the smartphone will upload them to a bulletin board. Notice that this generation of group elements is done only once in a while, and therefore can typically be performed when the smartphone is on charge and/or connected to a Wi-Fi network.

In Pronto-C2 we decouple the group element from the pseudonym by setting the 128 bit[9] pseudonym as the address on the bulletin board of the corresponding group element. In other words, a pseudonym is a pointer to a public memory, therefore one can just use a short string to refer to an arbitrarily large amount of data [10]. Recall that the infected person must compute the key K and push it to the server, while the non-infected person needs to compute the key K to then check if it

[9]This is the size for a pseudonym that is commonly allowed by BLE identifier beacons.

[10]A similar idea is used in IPFS https://ipfs.tech/.

exists on the server. Starting from a short pseudonym, every player will recover the corresponding group element from the bulletin board and use it to compute the key K.

Silent tracking. Pronto-C2 is clearly secure with respect to silent tracking. Indeed, when a person walks alone and passes by a silent tracking device, the sole transmission of the pseudonym used in that moment by the smartphone does not allow to understand if later on that person is infected. Indeed, the infected user's device will simply not upload any key K corresponding to a key agreement with a silent tracking device.

Shameless tracking. A government may also try to track citizens by having on its territory devices that behave as smartphones, broadcasting pseudonyms with the hope of receiving a call to infer some information on the locations and/or identities of the citizens. We refer to these tracking attacks as *shameless* tracking.

It goes without saying that such attacks are easier to detect compared to silent tracking. Indeed, the smartphone application could easily inform the owner at any time on the number of BLE identifier beacons that are being received. Therefore, there is more room for citizens to realize the existence of malicious devices. Any government that would like to save its reputation convincing citizens to still use the smartphone application should take severe actions against such tracking attempts. Notice that the only dangerous BLE devices are the ones that announce the very specific identifier beacon for the contact tracing system. There are specific codes to differentiate identifier beacons for different systems. Therefore, it is still completely fine to have on the territory devices (e.g., information kiosks) that use BLE to provide other services.

Pronto-C2 is also secure also against shameless tracking. Notice that with shameless tracking the infected user will upload a call for the active device that was in her proximity. However, there will be no way to link multiple calls coming from the same infected citizen[11].

[11]This holds when the uploaded calls cannot be linked through other side-channel information. We discuss this issue in the following paragraph.

Therefore, unless we are in the extreme case where there is only one new infected person in a large area and in a significant amount of time, Pronto-C2 protects infected citizens from attempts to track their movements through active BLE beacons.

Unlinkability over TCP/IP, timing, and other side-channel attacks. As in all ACT systems, users could be de-anonymized through the IP address when connecting to servers. Moreover, in Pronto-C2 when uploading a batch of group elements some attention should be paid so that they are not linkable. We therefore suggest the use of artificial delays and uploads of bogus data (i.e., dummy traffic) with the only purpose of making harder any profiling attempt. We also discuss a simple solution to mitigate the above linkability issues using mixers. By mixers we essentially mean proxy servers that, after batching and permuting several upload requests, submit them to the server hiding the users' IP addresses. We assume that each user can select her own favorite mixer among several of them belonging to heterogeneous entities (e.g., political parties, large organizations defending civil rights). By doing so, users could pick their favorite options to protect their IP addresses when uploading their pseudonyms and their anonymous calls to the bulletin board, and when downloading pseudonyms corresponding to the received BLE beacon identifiers. We give a more detailed description of this idea in Section 3.3.7. We remark that the server can perform mixing as well, and the privacy will be based on at least one among server and mixer behaving honestly. We stress that all ACTs are affected by such issues and mostly ignore them, but still we prefer to discuss possible workarounds, even though they obviously introduce extra overhead.

Countermeasures to DoS attacks. Typical DoS attacks can be mitigated with standard approaches, just to mention some: CAPTCHAs, proofs of work, anonymous tokens. We will discuss how to mitigate DoS attacks to the bulletin boards, by allowing regular (i.e., non-infected) citizens to upload a limited amount of pseudonyms.

3.3.1.2 Pronto-B2

While Pronto-C2 satisfies strong privacy notions, the mapping between pseudonyms and group elements introduces some overhead. We therefore propose also a lighter version of Pronto-C2 that we call Pronto-B2 and that does not require to translate the identifier beacon to a group element. Indeed, instead of performing a DH key exchange, Pronto-B2 performs a key exchange that is not confidential in the presence of an eavesdropper. This means that whoever is around Alice and Bob can compute the key K that Alice would use to make an anonymous call to Bob. The main observation is that if somebody is around, then she/he would also receive a call in case one out of Alice and Bob is infected, therefore the leakage is not that relevant. What mainly matters is that unlinkability holds. Indeed, calls made by an infected citizen corresponding to contacts in different locations and time slots must remain completely unlinkable (unlike in DP3T systems).

The key idea to construct Pronto-B2 is rather simple: let A_i and B_i be the 128-bit beacon identifiers used by Alice and Bob in some given epoch. If later on Alice is infected, she will make an anonymous call to Bob by uploading on the bulletin board $K = H(A_i||B_i)$ where H is a cryptographic hash function modelled as a random oracle. Notice that like in Pronto-C2, if Alice walks alone and there is a passive antenna, then no call is generated and Alice is not traced (unlike in DP3T). Notice also that if Alice is continuously in the range of antennas, but also in the presence of others, then the adversary would be able to see the calls generated in different epochs, but the adversary will not be able to link the calls[12].

3.3.2 Threat Model

In this section, we present the adversary goals, capabilities, and the threats we cover. We introduce several attacks against ACTs and for each of the attacks the adversary may have different goals, capabilities, and collude with different entities.

[12]Obviously this is true only when there is more than one infected citizens making calls, otherwise linking is always successful in any system.

Adversary goals and threats. An adversary attacking the privacy of the system wishes to *track* users. By tracking users we mean that the adversary can link different locations visited by the same user. This is irrespective of the fact that the adversary may not know the real identity of the user who visited such locations.

Notice that, due to the fact that pseudonyms in BLE packets do not change during a time slot, tracking in these periods is inevitable.

An adversary attacking the privacy also wishes to *link* locations visited by users, both diagnosed and non-diagnosed, to their real identities. For instance, the adversary may attack the privacy by attempting to link information uploaded by the user to a server through the user's IP address.

An adversary attacking the integrity wishes to falsely alert users of having been in contact with a diagnosed person; replay attacks fall into this category.

Adversary capabilities. We will consider different threats that involve adversaries with different capabilities. The adversary may place an arbitrary number of listening BLE devices at arbitrary locations. Such devices may operate exclusively in reception mode over the BLE channel. In this case, we say that the adversary is *passive* and is performing silent tracking. An adversary may also try to track citizens using devices that behave as regular smartphones. In this case, the adversary is *active* and is performing shameless tracking.

For some of our attacks we will consider an adversary that can even corrupt the server and/or the health authority.

Types of tracking attacks. Many of the proposed attacks (cfr., Sections 3.3.3.1, 3.3.3.2, 3.3.3.3) deal with tracking the movements of infected individuals over the contagious time window. Let us consider the strongest possible adversary who may try to track infected users, that is an adversary using active devices (i.e., behaving as regular smartphones) and colluding with the server. We now evaluate what is the highest level of privacy protection that can be guaranteed to infected users in this scenario. Note that an active adversary Adv is completely indistinguishable from a regular user of the system, this means that whenever Adv comes into contact with an infected user U who decides to upload data to the system, Adv will be alerted as prescribed by the system itself. In addition, Adv gets to see all the data uploaded by U to alert all the users she came into contact with. Suppose that Adv has placed a series of active devices over a certain territory, then for each of such locations where U has been over the contagious time window, U will upload data to server in order to alert Adv's devices. This means that Adv will certainly know which of his devices have been in proximity of an infected user and when.

Therefore, the best we can hope for in this scenario is that Adv cannot know whether data related to different locations are relative to the same individual. In this case, a certain degree of privacy is provided to infected users whose movements remain hidden within the set of movements of all other diagnosed people who uploaded data during the same day.

More specifically, we say that a protocol enjoys *partial protection* (w.r.t. passive or active adversaries) when an adversary who placed (passive or active) devices at two different locations X and Y cannot figure out whether X and Y have been visited by the same infected user.

Of course, in general, there may be additional information that helps Adv to disambiguate. For instance, consider the scenario in which a pseudonym is listened at location X at time t_1 and another one is listened at location Y at time $t_2 > t_1$. If no other pseudonym has been listened at nearby locations at times $< t_1$, there are two possible explanations: 1) a user turned on his phone at location X at time t_1 and then moved to position Y at time t_2; 2) a user U_1 turned on his phone at location X at time t_1 and then, at time t_2, U_1 turned his phone off while another user U_2 turned his phone on. In this simple scenario, it seems obvious that the first case is more likely than the second one. However, disambiguating gets more difficult as the number of infected individuals and locations increases.

We say that an ACT enjoys *full protection* from tracking attacks w.r.t. a passive adversary Adv, if Adv is not able at all to trace the movements of infected individuals during the contagious time window. To be more specific, even if there is a single infected user U, Adv does not get to know even one single location visited by U during the contagious time window.

Furthermore, U may pass nearby such devices both when she is alone or when some other users of the system are also there. In the first case, U may not upload any data related to the period of time she was alone since there is no one to be alerted, while in the second case an alert should be sent to whom has been in contact with U. For this reason, an ACT may exhibit different levels of resilience to tracking attacks depending on the actual encounters the user had, in particular: it could protect infected users from being tracked in any case (i.e., it provides full protection), or only for the periods of time they have been alone. We name the latter as *solitary protection* from tracking attacks. Obviously, *full protection* implies *solitary protection.*

Threats not addressed. We do not consider threats at the BLE layer such as using power signal and other side-channel information to identify users, or issues at the operating system level.

3.3.3 Privacy Attacks for Mass Surveillance

Mass surveillance is an activity put in place to watch, even discontinuously, over a substantial fraction of the population by monitoring, for example, their movements and/or habits.

Even though decentralized ACTs guarantee, in general, better privacy compared to centralized ones, mass surveillance is still a possible threat and must be mitigated as much as possible when introducing new intrusive technologies.

In the following paragraphs, we present several possible attacks to ACTs which, when successful, undermine users' privacy, eventually contributing to mass surveillance activities. Furthermore, we evaluate and compare the resilience of our Pronto-C2 and Pronto-B2 systems (cfr., Sec. 3.3.5) and the DP3T systems (cfr., Sec. 3.3.4.5) against such attacks.

Our attacks are inspired by the works of Vaudenay [115] and by the issues reported in the DP3T's GitHub repository. We carefully take into account these issues and attacks to illustrate more precise scenarios unveiling significant mass surveillance attacks.

Both the DP3T systems, Pronto-B2 and Pronto-C2 protect the privacy of non-diagnosed people and the privacy of diagnosed people out of the contagious time. Since these systems are the main focus of our work, in some of the following attacks we will assume that the adversary only attacks the privacy of diagnosed people during the contagious time window (e.g., roughly, the last two weeks before being tested positive). However, we remark that privacy protection is important for all users at any time, and ACTs might strongly violate privacy, especially when there is an extremely centralized design and when there is collusion among the various authorities of the system.

3.3.3.1 Paparazzi Attack: Tracking Infected Users with Trusted Server

This attack is similar to the paparazzi attack reported in [115]. The main difference between the two is that the one of [115] has the purpose of de-anonymizing infected users, while here we focus on building a mass surveillance infrastructure to track citizens.

- **Attacker's capabilities**: Adv has the ability to install, in a number of different locations, passive BLE devices. The only capability of a passive device is to operate over BLE channels in reception mode. We also assume that such devices are provided with enough memory to store a significant amount of received data (i.e., pseudonyms and auxiliary information).

- **Attack description**: The passive devices record the observed pseudonyms along with a fine-grained time log. The location of each device is fixed and determined by the attacker Adv. When a user B is tested positive and uploads data into the ACT system, the system itself provides related data to all users. Adv then combines these data with his logs. Furthermore, the attack is practically undetectable since the BLE devices operate only in reception mode.

- **Attack's outcome**: Adv tracks the infected users over the contagious time window. The attack is considered successful if the system fails to provide *full protection.* Variants of the attack can be considered, where Adv is interested in breaking *solitary protection* or *partial protection* respectively.

3.3.3.2 Orwell Attack: Tracking Infected Users with Colluding Server

Orwell attack differs from the paparazzi attack only for the capabilities of the attacker.

- **Attacker's capabilities**: The attacker Adv is the same as in paparazzi attack. However, in addition, Adv can collude with the server. Notice that the server could be under a significant influence of the government.

- **Attack description**: Adv is analogous to the one described in paparazzi attack. The only difference is that, along with data provided to all regular users, Adv receives all data that are in possession of the server.

- **Attack's outcome**: The outcome is analogous to the one of paparazzi attack. As in the paparazzi attack, we can consider variants where Adv is interested in breaking *solitary protection* or *partial protection* respectively.

3.3.3.3 Matrix Attack: Shameless Tracking of Infected Users with Colluding Server

- **Attacker's capabilities**: The attacker Adv is identical to the one of the Orwell attack in terms of the information he has access to. On the other hand, Adv's devices are active and can send messages over the BLE channel.

- **Attack description**: Adv operates similarly to what has been shown in the previous attacks. Adv combines the data in his possession with the ability to actively send messages of the contact tracing protocol over the BLE channel in order to track infected citizens over the contagious time window.

- **Attack's outcome**: Adv tracks the infected users over the contagious time window. The attack is considered successful if the system fails to provide *partial protection*.

3.3.3.4 Brutus[13] Attack: Mapping Real Identities to Pseudonyms

- **Attacker's capabilities**: The attacker Adv consists of the server and the health authorities colluding together.

- **Attack description**: Adv exploits the authorization mechanism, also used to avoid uploads of false positives, to find a mapping between the real identity of a user B and her uploaded data.

- **Attack's outcome**: a mapping between the real identity of B and her uploaded data.

Every ACT where the authorization mechanism consists of simply forwarding a token provided to B by the health authority is vulnerable to this attack. Indeed, the health authority - which is aware of the real identity of B - can communicate the mapping between the token and the real identity of B to the server, which can in turn derive the mapping between this token and data uploaded by B. The authorization mechanism is not made explicit in many proposals [94, 97]. A reason

[13]Marcus Junius Brutus was a close friend of Julius Caesar. Brutus had a leading role in his assassination. His name has become synonymous with severe acts of betrayal.

advocated for this choice is the flexibility to different deployment scenarios. However, we want to point out that the used authorization mechanism reflects into serious implications on users' privacy.

3.3.4 Other Attacks

3.3.4.1 Bombolo[14] Attack: Leakage of Contacts of Infected Users

- **Attacker's capabilities**: The attacker Adv consists of the server and the health authorities colluding together.

- **Attack description**: When users are tested positive, they upload data to the system. The attacker uses such data to extract information related to contacts among infected users and the number of contacts of an infected user.

- **Attack's outcome**: Adv succeeds in extracting information as specified in the attack description.

Systems in which the infected users upload an encoding of the observed pseudonyms are more prone to this attack since the content and the amount of communicated data depend on the actual number of experienced contacts. One could think to mitigate this issue by putting a bound on the number of contacts that a user can notify. However, it is not evident what is the appropriate value for this bound to effectively fight the pandemic.

Additionally, co-location of infected users is more likely to be exposed since infected users who met each other might end up reporting some linkable information. If at some point two infected users met each other, the information that these users sent to the server may enable the reconstruction of clusters of infected users who have been co-located. Nevertheless, such attacks are ineffective to link multiple

[14]Franco Lechner, best known as Bombolo, was an Italian comedian. His characters usually played hilarious but harmless jokes.

locations visited by a given user, and thus it is hard to imagine how such leakage could be exploited by mass surveillance attacks.

3.3.4.2 Gossip Attack: Proving Contact With an Infected User

This attack deals with the possibility to exploit the ACT in order to produce plausible digital evidence of an encounter. An attack of this type against the DP3T systems was reported by Pietrzak [96]. Here we give a description of such attack against a general ACT.

- **Attacker's capabilities**: The attacker Adv has the same power as a regular user. Additionally, Adv might get access to a service making him able to prove the ownership of some data at a specific time (e.g., a blockchain).

- **Attack's outcome**: Adv provides a plausible evidence of having met an infected user B before B declared himself as positive through the ACT.

Turning Gossip attack into a feature. Suppose that, due to the pandemic, laboratories are overwhelmed by requests for tests. In this scenario, having a way to prioritize the requests could be certainly useful. Indeed, there could be malicious users trying to fake risk notifications so that they eventually get tested, even if it is not actually needed.

To address this issue, one could leverage the Gossip attack as a feature. Laboratories could give a higher priority to users who are able to provide a plausible evidence of having met an infected individual. Depending on the system, a malicious user attempting to provide a fake proof would need the collaboration of someone who actually observed at least a pseudonym of an infected user. Such complications might reduce the noise of malicious users trying to create a fake plausible evidence. Therefore, prioritizing users with plausible (though not formally provable) evidence can be a concrete strategy for a health system.

3.3.4.3 Matteotti[15] Attack: Putting Opponents in Quarantine

- **Attacker's capabilities**: The attacker Adv colludes with the sever and the health authority. In addition, Adv can place passive BLE devices at selected locations.

- **Attack description**: The aim of Adv is to produce false alerts causing target non-at-risk users to get tested.

- **Attack's outcome**: A target non-at-risk user is erroneously alerted and declared as positive.

We motivate the attack with the following example. In the vast majority of world's countries e-voting is not currently deployed, and, also at parliamentary level, voting is always held in presence. Suppose that a law, proposed by the government, risks not to get the approval of the parliament for very few votes. A malicious government could attempt to falsely report hostile parliamentarians as positive.

Let B be a hostile parliamentarian. Hidden passive BLE devices could be put in place near the house of B during a given period. These BLE devices will intercept the pseudonyms $\mathsf{EphID_B}$s of B. Then, the government will add the $\mathsf{EphID_B}$s to the list of users that will be notified as at-risk users. It is very likely that the next day B will get tested. At this point, the test of B will come out as positive (the health authority is also malicious) and an order of quarantine for B would be issued. As a result, B would be unable to join the next parliament session.

3.3.4.4 Replay Attack

- **Attacker's capabilities**: The attacker Adv is anyone who is capable of recording and broadcasting pseudonyms.

[15]Giacomo Matteotti was an Italian socialist politician who openly denounced the electoral fraud committed by Fascists. He was kidnapped and killed by Fascists. The day he was murdered, Matteotti should have given a speech at the parliament in which he would have disclosed significant scandals about the Duce.

- **Attack description**: Vaudenay [115] and Pietrzak [96] discuss attacks in which Adv, who collects a pseudonym at location X where the probability to meet an infected person is high, can then broadcast those pseudonyms to users at a different location Y. We consider attacks where the listened pseudonyms are broadcast at a later time slot, and where messages are only relayed in one direction. We denote such attacks as replay attacks[16].

- **Attack's outcome**: Users at location Y will be notified a risk even though they have been never in contact with infected people at location X.

3.3.4.5 Security Analysis of the DP3T Systems

In this section, we analyse the security of both designs of the DP3T team w.r.t. the attacks proposed in Sec. 3.3.3 and 3.3.4.

The low-cost design of DP3T is vulnerable to Paparazzi attack. The attack is carried out as follows. The attacker Adv controls a set of passive devices $\{\mathsf{D}_1, \ldots, \mathsf{D}_n\}$.

1. Each passive device D_i collects the information of people that pass nearby D_i, the information stored consists of a set of pairs $(\mathsf{EphID}_j, \tau_j)$, where EphID_j is the pseudonym of a user that passes near D_i and τ_j is a fine-grained time log.
2. At the end of the day, Adv downloads the secret key of each infected user from the server and collects all data from each device D_i.

[16]If messages are relayed in real time and in both directions (i.e., back and forth between the likely infected users and the targets), then any solution inherently requires some location information (e.g., by GPS), or problematic assumptions on time synchronization. Therefore, we will focus only on replay attacks since they can be defeated without adding assumptions or penalizing privacy. Furthermore, replay attacks are very simple to carry out in practice, unlike real-time bidirectional relay attacks.

3. Adv checks if each collected EphID_j is generated starting by a secret key sk_j downloaded from the server.

4. Adv tracks the infected individuals who passed nearby the passive devices over a given contagious time window.

In the scenario we envision, the amount of gathered data can be considerably large, thus resulting in a possibly very fine-grained tracking.

The key issue of the low-cost design, leading to the applicability of paparazzi attack, lies in the fact that when the secret key of an infected person is added to the system everyone can derive all the related EphIDs, enabling the linking of several pseudonyms to the same infected individual. We point out that this attack is practically undetectable, at least at the application level, since the devices do not need to propagate any signal.

The unlinkable design of DP3T is vulnerable to Orwell attack. Since the Cuckoo filter allows users to only test inclusion of seemingly uncorrelated EphIDs in the filter itself, the unlinkable design succeeds in preventing the paparazzi attack. However, the claim that "infected people in the unlinkable design are not traceable", as affirmed in [110] is oversimplified and requires a deeper treatment. In fact, such claim is true only w.r.t. attackers who do not cooperate with the server. Considering an adversary that has control over the server, an attack analogous to the one described for the low-cost design can be put in place. In a nutshell, $\mathcal{A}$ obtains from the server all the seeds of the infected citizens and can now match the EphIDs recorded by his devices with the ones generated from the seeds received by the server. The element of centralization in the unlinkable design of DP3T requiring the server to compute the Cuckoo filter of the EphIDs enables mass surveillance with low overhead.

DP3T systems are vulnerable to Matrix Attack. Since both designs of DP3T are vulnerable either to the paparazzi or the Orwell attack, it clearly follows that they are vulnerable to the Matrix attack. In particular, the additional capability of placing active devices over the territory is not even needed by an adversary succeeding in the attack against both designs.

DP3T systems withstand Bombolo Attack. DP3T and similar systems are not affected by this attack. Indeed, the data which are sent to the server are independent on the actual encounters the infected user had.

DP3T systems are vulnerable to Brutus attack. DP3T proposes three candidate authorization mechanisms [112]:

1. Simple Authorization Codes: the server generates authorization codes that are distributed to infected users after a positive test;

2. Activated Authorization Codes: the authorization codes are assigned at testing time, and are activated only if the user tests positive;

3. Data-Bound Authorization: at testing time, the users commit the data to be uploaded to the server. Committed data is associated to an authorization code. If the user is tested positive, then the health authority authorizes the upload of the committed data associated to the authorization code.

It is simple to notice that the three mechanisms proposed by the DP3T team are subject to the attack. None of these mechanism addresses the problem of collusion between the server and the health authority. Indeed, the data uploaded by an infected user can be always related to a single authorization code, and then to the identity of the infected user.

DP3T systems are vulnerable to Gossip Attack. An attack of this type against the DP3T systems was reported by Pietrzak [96]. As plausible evidence of an encounter with a user B, A proves to have been in possession, at a time $t_1 < t_2$, of the pseudonym $\mathsf{EphID_B}$ of B who reported himself as positive to the ACT at time t_2. The attack is really straightforward. Whenever A receives a pseudonym from a user B, she commits it to a blockchain (e.g., Bitcoin). If B is later diagnosed infected and decides to upload his data to the system, A could then prove that she knew the pseudonym of B prior to this upload. To do so, A just needs to open the commitment on the blockchain. The attack works in the same way for both designs of DP3T, since the revealed $\mathsf{EphID}_\mathsf{B}$ can be easily matched both with the published filters and secret keys. Notice that there is no guarantee about the fact that A himself received the pseudonym over the BLE channel. For example, a device D in another (even remote) location could have committed the pseudonym and transferred its opening to A, by e-mail. However, in this case the attacker is actually the pair (A,D), who indeed met B. As noted in [96], the attack becomes a more serious threat if coupled with de-anonymization of B.

As we point out in Section 3.3.4.2, it is possible to consider this attack as a feature. Even though in the DP3T systems it is possible to provide a plausible evidence of being at risk by leveraging the Gossip attack as a feature, it seems, at least at a first glance, that it would not be easily scalable to a considerable portion of the users.

The DP3T white paper [114] proposes that users who are willing to do it may share additional data with epidemiologists. Such additional data are mainly related to encounters with infected individuals. Therefore, providing evidence of these encounters could help to ensure that data provided to the epidemiologists are reliable. Nevertheless, in both designs, DP3T does not provide a mechanism to verify the legitimacy of the alleged contacts.

The unlinkable design of DP3T is vulnerable to Matteotti attack. Even though the unlinkable design solves in part the issue of linkability of the pseudonyms, the attacker Adv that controls the server gets more power since Adv can add to the Cuckoo filter every EphID that Adv gets to know. This can cause additional false positives.

If Adv observes $\mathsf{EphID}_\mathsf{B}$ and $\mathsf{EphID}_\mathsf{C}$ in the same location and during the same time slot, then Adv can add $\mathsf{EphID}_\mathsf{B}$ and $\mathsf{EphID}_\mathsf{C}$ to the filter. The probability that, after checking the filter, both B and C are notified a risk is high since B will find $\mathsf{EphID}_\mathsf{C}$ in the filter as well as C will find $\mathsf{EphID}_\mathsf{B}$. Let us assume that B is the target of the attack. At this point, if B goes to a laboratory to get tested, the health authority would declare B as positive.

Replay attack against DP3T systems. According to [114], the low-cost design of DP3T is subject to replay attacks occurring within a day. Mitigations are proposed in [115] and [96]. On the other hand, since in the unlinkable design ephemeral identifiers are cryptographically linked to the time slot in which they are broadcast, it is in theory possible to make a replay attack only within a time slot. Differently to the low-cost design of DP3T, the pseudonyms in GAEN are implicitly tied to the time slot in which they are broadcast. Indeed, after they are derived from a TEK, the pseudonyms related to a given TEK are broadcast in a fixed order, instead of a random one. The documentation of GAEN [4] states that the system is vulnerable to replay attacks occurring in 2-hours windows.

Remark 1. The DP3T documents suggest 15 minutes as length of an epoch, concluding that in the unlinkable design the replay attack can be performed only in a time window of the same length. In our opinion, this is imprecise due to the following reason. To preserve privacy, the length of the epochs should be randomized otherwise trivial tracking attacks can be carried out. Therefore, the epoch should not be exactly 900 seconds but a random number of seconds between, e.g., 900-d and 900+d seconds, where d is a given tolerance (e.g., 60 seconds). Taking such randomization into account, the n-th ephemeral identifier derived from the secret key will be used up to $n \cdot d$ seconds later. Notice that randomization of epochs is concretely implemented in GAEN, and this seems to be the reason why GAEN declares itself vulnerable to replay attacks done in 2-hours-long time windows.

3.3.5 Pronto-B2 and Pronto-C2: Design and Analysis

One of the main drawbacks of previous solutions is the possibility for an attacker to test whether a set of pseudonyms belongs to the same infected person, and thus to infer the victim's movements. The problem is evident in the low-cost DP3T system but, as analyzed in Section 3.1, also arises in the DP3T's unlinkable variant.

Our approach diverges radically from the one of the DP3T systems in that we turn the paradigm upside down. In our system, the infected person is in charge of publishing data directly to people with whom he/she got in touch. It is up to each participant to verify the occurrence of a risk.

In this section, we present two protocols, Pronto-B2 and Pronto-C2. The former is more efficient but offers less privacy guarantees than the latter, still providing an arguably better protection than DP3T designs. We will describe Pronto-B2 omitting details regarding upload authorization and communication with the server that will be made explicit in Pronto-C2. Moreover, both in the description of Pronto-B2 and Pronto-C2 we will not explicitly deal with the use and the need of anonymous channels; these details will be taken in account in Section 3.3.7.

3.3.5.1 Pronto-B2

A brief overview. In a nutshell, Pronto-B2 works as follows.

Periodically, each user U performs the following update operation. Let i be the current time slot. U setups a set of ephemeral keys $\mathsf{Eph}_{\mathsf{U},i+j}, j = 0, \ldots, n-1$, for some parameter n; the ephemeral keys are random[17] strings of 128 bits. The idea is that these ephemeral keys will be used for the next n time slots. Every n time slots, U runs again the update operation, previous keys are not overridden.

At each time slot i, user U proceeds as follows. U broadcasts Eph_i and listens for ephemeral keys sent by other users. Every received key is recorded, along with auxiliary information.

Consider a simple scenario in which Bob tests positive and moreover he has been in close proximity to his neighbor Alice at time i (among possibly many other contacts). Let us denote by Eph_A (resp., Eph_B) Alice's (resp., Bob's) ephemeral key at the time of the contact. Bob sets $K = H(\mathsf{Eph}_\mathsf{B}||\mathsf{Eph}_\mathsf{A})$ where H is a cryptographic hash function, and uploads the "call" K to Server.

At the end of the day, to know whether she has been in contact with an infected person, Alice does the following. For each ephemeral key Eph_B she received from a nearby user at a time slot i, she computes $K = H(\mathsf{Eph}_\mathsf{B}||\mathsf{Eph}_\mathsf{A})$, where Eph_A is the Alice's ephemeral key at time i. She downloads from Server the recent calls and then searches for

[17]They can also be pseudorandom, to avoid delays in generating random bits.

occurrences of K in the downloaded calls. If K is present, she is notified the risk.

Pronto-B2's setting and actors. We assume the following:

- There is a server Server that is used as a bulletin board.
- The smartphone application has the capability to communicate with Server in an anonymous manner, hiding the real identity of the user.

The actors involved in our protocols are:

- The users who run a smartphone application endowed with a BLE identifier beacon. A generic user will be denoted by U.
- The server (Server) that manages the bulletin board.
- A set of medical laboratories (HAs) who are in charge of testing the users for the virus.

The Pronto-B2 system. Each user U keeps a set P_{U} that is empty at the onset. Moreover, U keeps an internal variable called *time slot.* At the start of the protocol U's time slot is set to 0 and each X seconds the time slot is increased by 1. X is a parameter of the protocol (e.g., 300 seconds).

We describe Pronto-B2 through the following procedures and events.

- Setup procedure. Each actor runs a setup as described in Fig. 3.1 when joining the system.
- Update procedure. This procedure, described in Fig. 3.2, is run periodically by each user U every n time slots (i.e., when U is at time slot j and j is a multiple of n)[18].

[18]We assume each time slot to be short enough to prevent significant linkage of ephemeral keys to users' movements, but long enough to correctly evaluate exposure risks. Moreover, we assume the parameter n to be sufficiently large to not require the users to perform the expensive Update procedure too frequently (e.g., n can be set so that the update is performed each week).

- U: configure the smartphone application and set the time slot to 0.
- Server: perform any necessary step to accept incoming requests.

Figure 3.1: Pronto-B2 Setup procedure.

In the Update procedure executed at time slot i, each user U does the following.

- U: for each $j = 0, \ldots, n-1$ generate an ephemeral key $\mathsf{Eph}_{\mathsf{U},i+j}$ drawing an element $\mathsf{Eph}_{\mathsf{U},i+j}$ at random from $\{0,1\}^{128}$.

Figure 3.2: Pronto-B2 Update procedure.

- Broadcast procedure. The Broadcast procedure, described in Fig. 3.3, is run multiple times within the time slot. The frequency with which this procedure is executed within a time slot is another parameter of the protocol.
- Listen Event. The Listen Event, described in Fig. 3.4, is triggered whenever a BLE identifier beacon is received.
- Test Positive Event. The Test Positive Event is triggered when a user tests positive at one of the laboratories of the HAs. Here, we ignore details about upload authorization and interaction with the HA that will be presented later in the description of Pronto-C2. After the test, U chooses a subset P'_{U} of P_{U}. U can decide upon which time slots to insert in P'_{U} based on any arbitrary criteria (e.g., she can exclude time slots in which U suspects to have met some people to whom she wants to hide his infection) and then perform an upload to Server. In more detail, when the event is triggered, U interacts with Server as depicted in Fig. 3.5.
- Verify procedure. This procedure, described in Fig. 3.6, is carried out by a user U who wants to discover whether she got in contact with some other user U^+ who tested positive.

- U: Let i be the current U's time slot. Broadcast via BLE the ephemeral key $\mathsf{Eph}_{\mathsf{U},i}$ generated in the last Update procedure.

Figure 3.3: Pronto-B2 Broadcast procedure.

When a BLE message is received as consequence of a broadcast procedure, the Listen Event is triggered by the user U that receives the message and proceeds as follows.

- U: let Eph_R be the received pseudonym, i the current time slot, and t any other auxiliary information (e.g., BLE signal, location, time). Add $(\mathsf{Eph}_{\mathsf{U},i}, \mathsf{Eph}_R, t)$ to the set P_U, where $\mathsf{Eph}_{\mathsf{U},i}$ is the ephemeral key that U computed in the last Update procedure.

Figure 3.4: Pronto-B2 Listen Event.

- Interaction between U and HA: once U is tested positive at HA, U is allowed to upload data to Server.

- $\mathsf{U} \rightarrow \mathsf{Server}$: choose a subset P'_U of P_U and for each triple $(\mathsf{Eph}_\mathsf{U}, \mathsf{Eph}_R, t) \in P'_\mathsf{U}$, compute $K = H(\mathsf{Eph}_\mathsf{U} || \mathsf{Eph}_R)$ and add K to K. H is a cryptographic hash function and K is the set of all calls that U wants to store on Server. Next, send K to Server.

Figure 3.5: Pronto-B2 Test Positive Event.

When a user U wants to verify whether she got in contact with any user U^+ who got a positive test result, U interacts with Server as follows.

- $\mathsf{U} \leftarrow \mathsf{Server}$: Let P_U be the set computed by U during the protocol execution so far. For each triple $(\mathsf{Eph}_\mathsf{U}, \mathsf{Eph}_R, t)$ in P_U, do the following:

    – Set $K = H(\mathsf{Eph}_R || \mathsf{Eph}_\mathsf{U})$, download the recently uploaded calls from Server and search for K. If K is present, compute the risk and notify U accordingly.

Figure 3.6: Pronto-B2 Verify procedure.

Remark 2. According to the above description, an infected user could listen for two ephemeral keys of other users and make a call using those ephemeral keys. The issue can be solved by requesting users to upload to the server a preimage of their ephemeral key w.r.t. some one-way function F. Obviously, in this case the value sent in broadcast will not be a (pseudo) random string but the output of F on input a (pseudo) random string. For simplicity, we omit such details. Henceforth, we will assume Pronto-B2 implicitly uses this slight modification.

Replay attacks. Pronto-B2, unlike DP3T systems, is resilient to replay attacks. Indeed, an adversary Adv who broadcasts, at location X, the pseudonym of a user U_1 collected during a prior time slot in a different location Y, would fail in the attempt of causing false at-risk notifications. This is because, to be notified, a user U_2 needs to find, on the bulletin board, a call which is directed to himself and is generated by the infected user U_1. However, if the two users never met, U_1 will not make calls containing a U_2's ephemeral key. Therefore, U_2 cannot receive a notification. Similarly, it is easy to see that Pronto-B2 is secure even against replay attacks that happen within the same time slot.

Tracking attacks in Pronto-B2. Our first protocol Pronto-B2 is not fully secure against Paparazzi and Orwell attacks (cfr., Sec. 3.3.3). Indeed, suppose that there is only one infected user. Then, an adversary who listened for the ephemeral keys broadcast by the users can use the calls uploaded to the server in order to trace the infected user. Indeed, if a call $(\mathsf{Eph}_1, \mathsf{Eph}_2)$ contains a given ephemeral key Eph_1 listened at location X and time t, this implies that the infected user who made the call has been at X at time t.

Nevertheless, if there are n infected users, each of them is hidden among the other infected individuals; this means that Pronto-B2 offers *partial protection* as described in Section 3.3.2. Indeed, the attacker may learn that some infected users visited two different places X and Y but he does not figure out whether the same infected user visited X and Y. This holds under the assumption that two calls appearing on the server cannot be linked to the same user, and that a call cannot be linked to a user. Furthermore, Pronto-B2 offers *solitary protection* as described in Section 3.3.2, while DP3T does not. That is, an infected user who has been at locations in which no other user was present cannot be traced. This intuitively follows from the fact that no calls are sent by infected users for time slots during which they had no encounters.

Moreover, unlike DP3T, Pronto-B2 is secure against Matrix attack. Recall that the adversary in Matrix can place active BLE devices. As discussed in Section 3.3.2, the best possible privacy guarantee regarding active attacks is *partial protection* and Pronto-B2 enjoys it. More precisely, under the assumption that different calls are unlinkable as explained above, even an active attacker cannot know which infected user visited a specific place out of all the possible infected users. The same holds for determining whether two different places have been visited by the same infected user.

Other attacks. Pronto-B2 (with the modification of Remark 2) is resilient to Matteotti attack. Indeed, every call K stored on the bulletin board has the form $K = H(\mathsf{Eph}_\mathsf{C}||\mathsf{Eph}_\mathsf{B})$ and additionally contains the preimage of Eph_C. A user B who at some time t broadcasts Eph_B will be notified a risk only if B received at time t an ephemeral key Eph_C and the preimage is consistent. Since it is hard for Adv to compute the preimage, we conclude that B is alerted only when B actually met C and C raised an alert for B.

Pronto-B2 is resilient to Brutus attack when a proper upload authorization mechanism is used. To this regard, more details can be found in the security analysis of Pronto-C2 (cfr., Sec. 3.3.6).

Regarding the gossip attack, Pronto-B2 is analogous to DP3T. Indeed, it is possible to carry out the same attack shown in Sec. 3.3.4.5.

Finally, Pronto-B2, has some resilience to Bombolo attack. Indeed, the calls consist of the output of a cryptographic hash function on the ephemeral keys so, under the random oracle assumption, it is hard for the attacker to get co-location information about infected individuals, even if the attacker is colluding with the server and the health authority. However, the number of contacts the user had is leaked to the health authority. To this regard, more details can be found in the security analysis of Pronto-C2 (see Sec. 3.3.6).

3.3.5.2 Pronto-C2

A brief overview. In a nutshell, Pronto-C2 works as follows. We assume the generator g of an elliptic curve group of prime order to be known to all participants.

Periodically, each user U performs the following update operation. Let i be the current time slot. U setups a set of ephemeral and secret keys $(\mathsf{Eph}_{\mathsf{U},i+j} = g^{\mathsf{sk}_{\mathsf{U},i+j}}, \mathsf{sk}_{\mathsf{U},i+j}), j = 0, \ldots, n-1$ for some parameter n. For $k = i, \ldots, i+n-1$, U sends to Server the string $\mathsf{Eph}_{\mathsf{U},k}$ and privately stores the address $\mathsf{addr}_{\mathsf{U},k}$ in which $\mathsf{Eph}_{\mathsf{U},k}$ appears on the bulletin board. The idea is that these addresses will be used for the next n time slots. Every n time slots, U runs again the update operation; previous pairs of ephemeral and secret keys are not overridden.

At each time slot i, user U proceeds as follows. U broadcasts addr_i and listens for addresses sent by other users. Each address received can be recorded along with auxiliary information.

Consider again the simple scenario in which Bob tests positive and he has been in close proximity to his neighbor Alice at time i. Let us denote by $\mathsf{Eph}_\mathsf{A} = g^{\mathsf{sk}_\mathsf{A}}$ (resp., $\mathsf{Eph}_\mathsf{B} = g^{\mathsf{sk}_\mathsf{B}}$) Alice's (resp., Bob's) ephemeral key at the time of the contact. Bob computes $K' = \mathsf{Eph}_\mathsf{A}^{\mathsf{sk}_\mathsf{B}}$ and uploads to Server the key (or "call") $K = H(K' || \mathsf{Eph}_\mathsf{B} || \mathsf{Eph}_\mathsf{A})$ after having required the authentication service AuthService to blind sign K. We use blind signatures to prevent the government to link users to information on the server. To perform the authentication, Bob needs to send to AuthService an activation code that Bob received from the laboratory when he got the diagnosis. We assume that Server accepts only keys with valid signatures.

At the end of the day, to know whether she has been in contact with an infected person, Alice does the following. For each address she received from a nearby user, she retrieves from Server the corresponding ephemeral key. As a result, she has Bob's ephemeral key Eph_B. She computes $K' = \mathsf{Eph}_\mathsf{B}^{\mathsf{sk}_\mathsf{A}}$ and $K = H(K' || \mathsf{Eph}_\mathsf{B} || \mathsf{Eph}_\mathsf{A})$, downloads from Server the recent calls, and then searches for occurrences of K in the downloaded calls. If K is present, she is notified the risk.

As additional step to avoid DoS attacks, we add a further authentication step when users store their ephemeral keys to the bulletin board: each user U that wants to store the ephemeral keys to the bulletin board must contact AuthService to obtain a blind signature for each of the ephemeral keys that U wishes to store. This step will leak information to AuthService: AuthService will know which person is using Pronto-C2 and which one is not using it. It is possible to mitigate this issue by requiring persons not using Pronto-C2 to request to AuthService blind signatures for dummy ephemeral keys.

Pronto-C2's system and crypto ingredients. The ingredients of our system are:

- An elliptic curve group of prime order p where the DH key exchange is believed to be secure. We assume a generator g of the group to be publicly known to all participants.
- A blind signature scheme. The blind signature is used to make AuthService sign users' data while hiding the signed data. We refer to [41, 42] for the syntax and security properties of blind signatures.
- A server Server that is used as a bulletin board. The server allows a user to write data iff the user is able to provide a valid (blind) signature issued by the authentication service.
- We assume the smartphone application has the capability to communicate with Server in an anonymous manner, hiding the real identity of the user.

Pronto-C2's setting and actors. The actors involved in our protocols are:

- The users who run a smartphone application endowed with a BLE identifier beacon. A generic user will be denoted by U.
- The server (Server) that manages the bulletin board.
- A set of medical laboratories (HAs) that test users for the virus and release the activation codes to infected users (see below).

- The authentication service (AuthService) that is used by all users to be authorized to upload the ephemeral keys on the bulletin board, and by infected users to get the authorization to write data about contacts on the bulletin board. AuthService releases a set of random activation codes to each HA. A user U is handed an activation code Code from HA when tested positive. U can later use Code to request a signature on some data K to AuthService. The authentication service will blind sign K only if Code is a valid (not previously used) authentication code released by AuthService. U can then use the signature to upload K to Server.

The Pronto-C2 system. Each user U keeps a set P_{U} that is empty at the onset. Moreover, U keeps an internal variable called *time slot*. At the start of the protocol U's time slot is set to 0 and each X seconds the time slot is increased by 1. X is a parameter of the protocol (e.g., 300 seconds).

We describe Pronto-C2 through the following procedures and events.

- Setup procedure. Each actor runs a setup as described in Fig. 3.7 when joining the system.

- Update procedure. This procedure, described in Fig. 3.8, is run periodically by each user U every n time slots (i.e., when U is at time slot j and j is a multiple of n)

- Broadcast procedure. There is a Broadcast procedure, described in Fig. 3.9 that is run multiple times within the time slot. The frequency with which this procedure is executed within a single time slot is another parameter of the protocol.

- Listen Event. The Listen Event, described in Fig. 3.10, is triggered whenever a BLE identifier beacon is received.

- Test Positive Event. The Test Positive Event is triggered when a user tests positive at one of the laboratories of one of the HAs. When a user U gets a positive test, U gets from HA an activation code Code. After the test, U chooses a subset P'_{U} of

- U: configure the smartphone application and set the time slot to 0.
- Server: perform any necessary step to accept incoming requests.
- AuthService: publish two public keys for a blind signature scheme, one to be used by the users to request an authorization to upload ephemeral keys and, one to request an authorization to upload shared keys. In addition, choose random activation codes and distribute a set of them to each HA.
- HA: receive a set of activation codes from AuthService.

Figure 3.7: Pronto-C2 Setup procedure.

P_{U}. U can decide upon which time slots to insert in P'_{U}. U interacts with AuthService to get the blind signatures and then perform an upload to Server. More in details, when the event is triggered, U interacts with Server, AuthService, and HA as depicted in Fig. 3.11.

- Verify procedure. This procedure, described in Fig. 3.12, is carried out by a user U who wants to discover whether she got in contact with some other user U^+ who tested positive.

3.3.6 Analysis of Pronto-C2

In this section, we informally show that Pronto-C2 withstands all the attacks shown in Section 3.3.3 and 3.3.4. For the sake of simplicity, we analyze Pronto-C2 as if ephemeral keys were directly sent over the BLE channel, ignoring the use of addresses. We therefore assume that the procedure to store ephemeral keys on the bulletin board effectively hides the real identity of the owner of each key. In the informal analysis of the paparazzi and Orwell attacks, we also consider the case of an adversary who is able to link all the calls of an infected user. We do that to remark that security w.r.t. the paparazzi and Orwell attacks holds regardless of the channel used by the infected users to upload data to the Server.

In the Update procedure executed at time slot i, each user U interacts with Server and $\mathsf{AuthService}$ as follows.

- $\mathsf{U} \rightarrow \mathsf{AuthService}$: for each $j = 0, \ldots, n-1$ generate a pair of ephemeral and secret keys $(\mathsf{Eph}_{\mathsf{U},i+j} = g^{\mathsf{sk}_{\mathsf{U},i+j}}, \mathsf{sk}_{\mathsf{U},i+j})$ drawing an element $\mathsf{sk}_{\mathsf{U},i+j}$ at random from $\mathbb{Z}_p$.[a] For each $j = 0, \ldots, n-1$ interact with $\mathsf{AuthService}$ to obtain a blind signature of $\mathsf{Eph}_{\mathsf{U},i+j}$.
- $\mathsf{U} \rightarrow \mathsf{Server}$: for each $j = 0, \ldots, n-1$ upload $\mathsf{Eph}_{\mathsf{U},i+j}$ to Server along with the corresponding blind signature computed in the previous step, and store the address addr_{i+j} in which $\mathsf{Eph}_{\mathsf{U},i+j}$ appears on the bulletin board.

[a]To optimize the space, the user could choose a single seed s during the Setup procedure and in each time slot i derive $\mathsf{sk}_{\mathsf{U},i} = \mathsf{PRF}(s, i)$.

Figure 3.8: Pronto-C2 Update procedure.

- U: Let i be the current U's time slot. Broadcast via BLE the address addr_i generated in the last Update procedure.

Figure 3.9: Pronto-C2 Broadcast procedure.

When a BLE message is received as consequence of a broadcast procedure, the Listen Event is triggered by the user U that receives the message and proceeds as follows.

- U: let addr_R be the address contained in the received message, i the current time slot, and t any other auxiliary information (e.g., BLE signal strength, location, time).

  Add $(\mathsf{Eph}_{\mathsf{U},i}, \mathsf{sk}_{\mathsf{U},i}, \mathsf{addr}_R, t)$ to the set P_{U}, where $\mathsf{Eph}_{\mathsf{U},i}$ (resp., $\mathsf{sk}_{\mathsf{U},i}$) is the ephemeral key (resp., secret key) that U computed in the last Update procedure.

Figure 3.10: Pronto-C2 Listen Event.

- Interaction between U and HA: once U is tested positive at HA, U gets from HA an activation code Code to interact with $\mathsf{AuthService}$.

- $\mathsf{U} \leftarrow \mathsf{Server}$: U chooses a subset P'_{U} of P_{U} and for each quadruple $(\mathsf{Eph}_{\mathsf{U}}, \mathsf{sk}_{\mathsf{U}}, \mathsf{addr}_R, t) \in P'_{\mathsf{U}}$, retrieves from Server the ephemeral key Eph_R stored at address addr_R. U computes $K' = \mathsf{Eph}_R^{\mathsf{sk}_{\mathsf{U}}}$ and $K = H(K'||\mathsf{Eph}_{\mathsf{U}}||\mathsf{Eph}_R)$ and adds K to K, where K is the set of all keys that U wants to store on Server. Next, the following steps are performed:

  - Interaction between U and $\mathsf{AuthService}$: for each value $K \in \mathsf{K}$ computed by U as before, U uses its activation code Code to interact with $\mathsf{AuthService}$ to compute a blind signature σ of K.
  - $\mathsf{U} \rightarrow \mathsf{Server}$: for each $K \in \mathsf{K}$ computed by U as before, send K and σ to Server.
  - $\mathsf{Server} \leftarrow \mathsf{U}$: upon receiving any pair (K, σ) from U, verify σ and if the signature is valid add K to the bulletin board.

Figure 3.11: Pronto-C2 Test Positive Event.

When a user U wants to verify whether she got in contact with any user U^+ who tested positive, U interacts with Server as follows.

- $\mathsf{U} \leftarrow \mathsf{Server}$: Let P_{U} be the set computed by U during the protocol execution so far. For each quadruple $(\mathsf{Eph}_{\mathsf{U}}, \mathsf{sk}_{\mathsf{U}}, \mathsf{addr}_R, t)$ in P_{U} do the following:

  - Retrieve from Server the ephemeral key Eph_R located at address addr_R. Compute $K' = \mathsf{Eph}_R^{\mathsf{sk}_{\mathsf{U}}}$ and $K = H(K'||\mathsf{Eph}_R||\mathsf{Eph}_{\mathsf{U}})$, download the recently uploaded keys from Server and search for K. If K is present, compute the risk and notify U.

Figure 3.12: Pronto-C2 Verify procedure.

- Paparazzi attack: Recall that this attack assumes that the attacker Adv uses only passive devices which operate in reception mode and are not able to transmit any signal. The only information a passive device D observes consists of ephemeral keys exchanged by users at the position in which D is located. Adv could try to track an infected user U exploiting the calls available on the bulletin board. Since the devices used by Adv are passive, no calls of U will ever be directed to a device D controlled by Adv. The only way for Adv to track U is to extract the ephemeral keys used to generate the calls and associate them to a single user. Since the calls are anonymously sent to Server, it is impossible for Adv to link together the calls of U. However, the analysis of Orwell attack reported below shows that, even linking all the calls, it would be impossible for Adv to extract the ephemeral keys related to each call.

- Orwell attack: The attack differs from the paparazzi attack in the fact that the adversary Adv can collude with Server, AuthService and HA. In this scenario, Adv has the following advantages w.r.t. the attacker of the paparazzi attack:

  – Adv knows the real identities of the infected users and the associated activation code Code assigned to them by HA;

  – Adv knows the blinded messages that the infected users asked AuthService to sign.

  This information is not useful for Adv to track a user U. Indeed, to track U, Adv needs to link all the calls published by U with U's ephemeral keys. Assuming that the upload of the calls on the bulletin board is performed through an anonymous channel, in order to link these calls, Adv needs to discover which calls was blinded by U to obtain the corresponding signature from AuthService. This would require Adv to break the blindness property of the blind signature, that is unlikely for a polynomial adversary. Even if Adv somehow managed to link all the calls (e.g., the channel used to upload data is not anonymous), the additional information received is the set of calls sent to Server by U, but none of them can be a call that Adv can understand, since none of passive devices controlled by Adv is the recipient of a call. Then, Adv would need to take all the couples of ephemeral keys recorded by each passive device D_i, and try to compute a call between the two users that own these two ephemeral keys. If the computed call is equal to a call published by U, then Adv knows that U was located in proximity to D_i. By doing it for each passive devices, Adv could track the movements of U in the contagious time window. However, if Adv is able to successfully compute a call starting from two ephemeral identifiers, it is easy to show that Adv can be used to define an adversary breaking the computational Diffie-Hellman assumption.

- Matrix attack: Assuming that an anonymous channel is used for each upload, it is straightforward to see that Pronto-C2 withstands the Matrix attack. Indeed, as already argued before, Adv cannot link U's calls in different time slots since U uses different and unlinkable pseudonyms that in turn will produce unlinkable DH keys. Pronto-C2 offers *partial protection* to tracking attacks. Adv will not be able to link multiple calls to the same infected user, hence the movements of an infected user remain hidden within the movements of the other infected users. Obviously, the actual location related to the different calls may act as another information to disambiguate the path of an infected user, but this is inherent for such a strong adversary (cfr., Sec. 3.3.2). Nevertheless, since Pronto-C2 provides *partial protection*, the adversary gets lost as soon as there is some ambiguity.

- Bombolo attack: Co-location information among infected users is not leaked since an infected user A will upload to Server a call of the form $K_{\mathsf{A}} = H(\mathsf{Eph}_{\mathsf{B}}^{\mathsf{sk}_{\mathsf{A}}}||\mathsf{Eph}_{\mathsf{A}}||\mathsf{Eph}_{\mathsf{B}})$ if A passed nearby B, likewise if B is an infected user B will upload the key $K_{\mathsf{B}} = H(\mathsf{Eph}_{\mathsf{A}}^{\mathsf{sk}_{\mathsf{B}}}||\mathsf{Eph}_{\mathsf{B}}||\mathsf{Eph}_{\mathsf{A}})$ if B passed nearby A. Then, the calls K_{A} and K_{B} uploaded by A and B are different and hard to "co-locate"[19]. On the contrary, Pronto-C2 exposes the number of calls that an infected user U does when U sends these calls to the AuthService. We notice that this leak of information can be

[19]This is easy to see if we think of H as a random oracle.

mitigated adding some dummy calls.

- Brutus Attack: The data uploaded by a user U in $\mathsf{Pronto\text{-}C2}$ cannot be linked to the real identity of U. Data are uploaded in the following steps:

  1. when the infected user uses Code to access $\mathsf{AuthService}$ in order to obtain the blind signatures of the calls and
  2. when the infected user uploads the calls to Server along with the unblinded signatures.

  The first step involves uploading Code to $\mathsf{AuthService}$ in order to obtain the blind signature of the calls. Since HA knows the real identity of each infected user, it is possible for Adv to link the blind signature requests with an infected person. However, since the upload of the calls is performed through an anonymous channel, Adv cannot link the calls with the signature requests thanks to the blindness property of the signature scheme. This of course hides the authorship of the uploaded data only inside the set of the infected users, which is known to Adv.

- Gossip attack: At first sight, one could think that a proof of contact with an infected user U can be given by user A providing a proof about the calls on the bulletin board. For instance, let A be a user holding a secret key $\mathsf{sk_A}$ corresponding to Eph_A and let $\mathsf{Eph_U}$ be the ephemeral key of a user U. If A finds a call $K = H(\mathsf{Eph_U^{sk_A}}||\mathsf{Eph_U}||\mathsf{Eph_A})$ on the bulletin board, A could prove that he knows the secret key $\mathsf{sk_A}$ corresponding to $\mathsf{Eph_A}$ and that K is computed as before, thus proving that U made a call to A. However, recall that in $\mathsf{Pronto\text{-}C2}$ all the pseudonyms used by the users are made public on the bulletin board. So, A could use $\mathsf{Eph_U}$, that is public on the bulletin board, to compute a call $K = H(\mathsf{Eph_U^{sk_A}}||\mathsf{Eph_U}||\mathsf{Eph_A})$ (a call from U to A) and show $\mathsf{sk_A}$ as proof of the fact that such call has been done. Generally, any proof of the fact that U made a call to A is *not* evidence of the fact that U met A since such proof could have been computed by A even if U has never been in contact with A. To do so, A needs to be marked as infected. On the other hand, if A is not infected, a proof of the fact that U made a call to A is instead plausible evidence of the fact that U met A. For this reason, we say that the attack affects Pronto-C2 minimally in the sense that an attacker A can provide a proof (i.e., the secret key $\mathsf{sk_A}$) of the contact between U and A that convinces a third party B who believes that A is not infected. Using this fact, we can interpret this attack as a *feature*. Indeed, as stated in Section 3.3.4.2, laboratories could give a higher priority to users who are able to provide such a plausible evidence of having met an infected individual.

- Matteotti attack: Every key K stored on the bulletin board has the form $K = H(K'||\mathsf{Eph_C}||\mathsf{Eph_B})$. A user B who at some time t broadcasts $\mathsf{Eph_B}$ will be notified a risk only if B received at time t an ephemeral key $\mathsf{Eph_C}$ and $K' = \mathsf{Eph_C}^{\mathsf{sk_B}}$. Since it is hard for Adv to compute K' without knowing $\mathsf{sk_B}$ or $\mathsf{sk_C}$, we conclude that B is alerted only when B actually met C and C put an alert for B. However, in such case the alert corresponds to an actual risk for B and does not represent a successful attack.

- Replay attack: Pronto-C2 is not subject to replay attacks. An adversary Adv who broadcasts, at location X, the pseudonym of a user U_1 collected during a prior time slot in a different location Y, would fail in the attempt of causing false at-risk notifications. Indeed, to be notified, a user U_2 needs to find a call which is directed to himself and is generated by the infected user U_1. However, user U_1 would never make such a call. Similarly, it is easy to see that Pronto-C2 is secure even against replay attacks that happen within the same time slot.

In Tab. 3.1, we report a recap comparison among our ACTs and DP3T's ones.

3.3.7 Suggestions for a Practical Realization of our ACTs

In this section, we give some suggestions for a practical implementation of Pronto-B2 and Pronto-C2. In particular, we focus on linkabil-

Attacks	Low-cost DP-3T	Unlinkable DP-3T	Pronto-C2	Pronto-B2
Paparazzi	✗	✓	✓	†
Orwell	✗	✗	✓	†
Matrix	✗	✗	✓	✓
Bombolo	✓	✓	◯	◯
Brutus	✗	✗	✓	✓
Gossip	✗	✗	◯	✗
Matteotti	✓	✗	✓	✓
Replay	✗	✓	✓	✓

Table 3.1: We show which system is susceptible to which attack. ✗ denotes that the system is vulnerable to the attack, ✓ denotes safety against the attack, ◯ denotes almost safety against the attack in the sense that the system is only very mildly affected (cfr., Section 3.3.6), and † denotes that the protocol offers *solitary protection* as described in Section 3.3.2. Recall that Gossip attack can be easily turned into a feature in Pronto-C2, but not in DP3T systems or in Pronto-B2.

ity/deanonymization attacks due to timings and IP addresses of the TLS connections with Server when uploading or downloading data. Such attacks also affect the DP3T designs that seem to ignore them, and in general are applicable to any system if no specific countermeasure is used.

A mitigation based on mixers. One might consider onion routing to protect U against such attacks, but it is unclear whether the impact on the performance would make the system unpractical. In order to give a fair idea on a practical realization of our solutions, we do not ignore this issue. Therefore, we propose a mitigation based on simple proxy servers we call mixers. We consider a setting in which U can freely select a mixer MixServer that she trusts. U will use MixServer as a proxy to upload data to (or download data from) Server. The selected mixer is trusted only for keeping the IP address of the user secret. If a mixer does not perform the upload, this can be easily detected and the user can delegate the upload to another mixer.

Who is running the mixers? Mixers do not need to be approved by the government and they can be spontaneously run by anyone. There can be several heterogeneous mixers available provided by large institutions like no-profit organizations, political parties, national/state/local governments, as well as smaller mixers that can serve a community. U will choose the one that he trusts more in performing properly the service with a sufficiently large amount of collected data.

High-level workflow. Server owns a pair of private and public keys $(\mathsf{sk}_{\mathsf{Server}}, \mathsf{pk}_{\mathsf{Server}})$ of a public key encryption scheme, the public key of Server is made publicly available at set-up time. Every time U has to send data to Server, U will actually encrypt the data with $\mathsf{pk}_{\mathsf{Server}}$ and send the resulting ciphertexts to $\mathsf{MixServer}$. A mixer waits for enough data to be collected, and then permutes it and sends it to Server. In addition, $\mathsf{MixServer}$ can download all data from Server so that U can use $\mathsf{MixServer}$ also to retrieve anonymous calls and ephemeral keys. Server works as a bulletin board, so all data ever received by Server are made publicly available after being decrypted. If the user feels uncomfortable in sharing her infected status with $\mathsf{MixServer}$, she can periodically send to $\mathsf{MixServer}$ some dummy calls to be sent to Server. After decryption, Server will discard such dummy calls. It remains possible for a user to ignore this suggestion sending the encrypted data directly to Server using onion routing, and/or relying on the partial hiding provided by mobile operators and public Wi-Fi networks.

3.3.8 Disproving some DP3T's Claims

Aside from providing alternative ACTs with improved integrity and privacy guarantees, $\mathsf{Pronto\text{-}B2}$ and $\mathsf{Pronto\text{-}C2}$ also act as concrete examples against some claims made by the DP3T team to justify the security of their systems. In particular, regarding privacy, we have focused on mass surveillance attacks since they were one of the reasons advocated to prefer the decentralized approach. We now report the claims that are disproved (or at least downsized) by our ACTs.

Privacy attacks. We analyze the following claims related to privacy attacks:

- In [110], the following is presented as an inherent risk of ACTs (IR1): "any proximity tracing system that notifies users that they are at risk enables a motivated attacker to identify the infected people that he or she has been physically near. This risk is a consequence of the basic proximity tracing functionality." Although this claim is not incorrect per-se, it may lead to wrong conclusions. Indeed, the DP3T team answered to Vaudanay's paparazzi attack [111] referencing the above claim to convey that the paparazzi attack was inherent to all ACTs. Nevertheless, as we extensively showed, this is simply not true since both Pronto-C2 and Pronto-B2 withstand the paparazzi attack. As we argued in Sec. 3.3.2, such claim is true w.r.t. an active adversary since the best we can aim for in that case is *partial protection*, and that does not prevent de-anonymization. Indeed, it only guarantees that locations of infected users cannot be linked. Nevertheless, those locations will be revealed since the attacker is de-facto indistinguishable from a regular user who has to be notified. However, the adversary of the paparazzi attack is much weaker as it only uses passive devices, and thus she does not behave like a regular user. In conclusion, we agree that de-anonymization of infected users is unavoidable if we consider the most powerful adversary but we remark that such fact should not be used to deem as inherent every possible de-anonymization attack, even the ones carried out by very weak adversaries. Moreover, this discussion also shows that DP3T leaks strictly more of what is required by the proximity tracing functionality which should imply be alerting the at-risk users only, and not any curious spy eavesdropping the channel without declaring his presence.

- In [110], when discussing tracking of infected individuals (SR4), the following claim is made "infected people in the unlinkable design are not traceable". Here, as opposed to what was done in the previous point, the weakest possible adversary is implicitly considered. Namely, a passive adversary who does not collude with the server. However, if the adversary just colludes with the server (as it would happen in mass-surveillance scenarios) while still remaining passive, the above claim is false. We have shown why in the analysis of the Orwell attack in Sec. 3.3.4.5.

- In [110], the DP3T team analyzes ACTs where “an infected person uploads all identifiers observed during the contagious window to the server”. They argue that “for epochs in which groups of at least three people were in close proximity to each other, this will reveal temporal colocation information about infected individuals to the server”. This claim is true when systems uploading recorded pseudonyms are implemented in the most basic way, having the smartphones simply uploading the pseudonyms they recorded over the BLE channel as they are. If two smartphones listened the same pseudonyms, they would upload the same data, allowing the server to easily infer co-location information about infected users. This risk is modeled in our work through the Bombolo attack. We showed with both Pronto-B2 and Pronto-C2 that it is possible to not expose any colocation information while still uploading data about the encounters infected users had during the contagious time window. We achieve this without any impact on performance (i.e., just using a hash function), by observing that there are other solutions rather than simply forwarding to the server the listened pseudonyms. The same techniques can be equally applied to centralized systems, where the exposure of colocation information of users (also known as the social graph) was presented as a serious privacy concern (see SR8 of [110]).

Integrity attacks. Regarding integrity attacks, the risk of false alarms through active relays is discussed in [110] (GR2). In this scenario, the attacker tries to create fake alerts by simply relaying interactions with devices that are likely to be marked as infected later on. In [110], solutions to these attacks are deemed possible only via distance bounding protocols or by binding transmissions to locations. This claim conveys the idea that replay attacks are unavoidable unless too complicated techniques are used, or users' privacy is reduced. However, as we demonstrated, this is true only for the very powerful adversary that relays messages in real time and back and forth between the likely infected users and the targets. Indeed, both Pronto-C2 and Pronto-B2 are resilient to replay attacks (where relays are unidirectional and can happen at later time slots) without relying on any of the strong assumptions reported above. We also remark that in our systems even the most powerful adversary would have a hard time trying to perform a large-scale attack without being detected. Indeed, for the attack to be successful the infected users would have to upload a big number of calls even if they were not in proximity to a big number of people. This allows to detect the attack at the protocol level, concretely preventing its success. On the other hand, in DP3T/GAEN the attacker could replay pseudonyms over a stadium with thousands of people, and the attack could be detected only after the fact, when many people request to get tested after having received a notification. At this point the attack could already be considered a success. Indeed, the goal of the adversary could just be jeopardizing the health system of a region/country.

3.3.9 Provable Security of our ACTs

In Sec. 3.3.6, we informally evaluated the security of our system w.r.t. the attacks we elucidated in Sec. 3.3.3 and Sec. 3.3.4. The main intent of our work was to better clarify the state of affairs about the security guarantees that ACTs can offer, demonstrating that basic decentralized designs are not inherently eliminating serious mass surveillance attacks, as well as proposing alternative approaches.

Few months after our work [12] was made available on ePrint, a paper by Canetti et al. [34] focusing on formulating security definitions for ACTs was published on ePrint. They also proposed an ACT called CleverParrot whose design is pretty similar to Pronto-C2. The main difference is that CleverParrot does not use addresses but it directly broadcasts DH keys over the BLE channel. As a result, if we assume the upload channel to the server to preserve users' anonymity, their analysis directly applies to Pronto-C2 as well. We can match some of their security notions to some of our attacks, thus confirming our analysis. In particular, their notion of diagnosis listener privacy implies resilience to the paparazzi attack, their notion of upload unlinkability is analogous to resilience to the matrix attack, and their notion of contact provability is analogous to modelling the gossip attack as a feature. As integrity guarantee, they also model resilience to replay attacks. Finally, their notion of mass notification limits addresses the detectability problem of large-scale relay attacks that we mentioned in the previous section. For more details about their systems and their security notions, we refer the reader to [34].

3.4 False Alerts Injection: the Terrorist Attack

Since BLE was not originally designed to detect a precise distance among devices, the evaluation of the risk factor is prone to significant errors. To this regard, Leith and Farrell recently evaluated the reliability of BLE for digital contact tracing in several real-world scenarios [84].

While false positives due to BLE limitations in measuring distance can indiscriminately affect all individuals using the smartphone apps, a much more concerning threat allowing to direct false positive alerts to specific targets are replay attacks. Indeed, GAEN-based contact tracing systems[20] can be abused through replay attacks [67, 96, 115]. In [99], Gennaro et al. discussed how the capability of running such attacks at large scale can be used to put a category of citizens in quarantine with the consequence of severely compromising the results of an election. In general, the malicious generation of false positives can be harmful in various ways, the health system can be overloaded of requests that can penalize those citizens who instead are really affected by the virus. A student can cause the complete closure of a school or university and similar attacks can be directed to shops, malls, gyms, post offices, restaurants, or factories.

Risks related to replay attacks were already known back in April 2020, and GAEN systems have a pretty large time window (about 2 hours) for pseudonyms to be replayed successfully [67]. Nevertheless, governments have so far considered unlikely that such attacks can produce enough damage to cancel out the positive effects of genuine

[20]Sometimes we will refer to them simply as GAEN systems.

notifications of at-risk contacts. This could be due to complications involved in the attack. Indeed, an adversary may not want to get herself infected, or it could not be easy to identify - and be in physical proximity with - an individual that soon will report to be infected.

In [116], Vaudenay envisioned the possibility of using smart contracts to realize a *terrorist* attack against decentralized systems. Therefore, the attack could potentially apply to GAEN-based systems as well. In this case, the attacker would spread on his targets some pseudonyms, subsequently promising through a smart contract a reward to whoever uploads the corresponding keys. Therefore, an infected individual who participates in the contract will cash a reward, and false positive alerts will raise on the smartphones of the targets selected by the terrorist.

Dissemination of infected TEKs in GAEN-based ACTs In GAEN, the TEKs of infected users are disseminated via a back-end server that periodically posts a list of digitally signed TEKs. An important point is that the user's device evaluates the reported TEKs if and only if the digital signature verifies successfully under a public key that has been previously communicated by the developers of the ACT to Apple and Google. Google motivates this requirement saying that it ensures that keys received by the devices are actually from the authorized server and not from malicious third parties[21]. Theoretically, one could also rely on server authentication using TLS, but the use of Content Delivery Networks (CDNs) to disseminate TEKs (e.g., the CDN used by Immuni is operated by Akamai, while the SwissCovid's one is operated by Amazon) requires protection against malicious modifications operated by the CDN itself. Unfortunately, as we will see next, this requirement paves the way for the development of dark economies where TEKs to be uploaded by infected users are traded through smart contracts.

[21]Google: Exposure Notification Reference Key Server https://google.github.io/exposure-notifications-server/.

3.4.1 Terrosist Attack: Our Contribution

In this thesis, we show that the terrorist attack envisioned by Vaudenay can be concretely mounted against currently deployed GAEN-based contact tracing systems. In particular, we have analyzed its concrete feasibility w.r.t. two systems, such as Immuni [78], used in Italy and SwissCovid [105], used in Switzerland. We expect several other deployed GAEN systems to suffer from the same vulnerabilities.

More generally, our work shows how to attack the integrity of currently deployed GAEN-based contact tracing systems by leveraging blockchain technology. A very alarming side of our contribution is that current systems can be compromised without the need for the attacker to get infected, or to be with high probability in close proximity to individuals that will be soon detected positive and will upload the keys. Our attacks consist of smart contracts to establish a mediator-free market where parties, without knowing each other, without meeting in person and without running risks to be cheated, can abuse exposure notifications procedures of GAEN systems. We give a brief description of the mentioned smart contracts in the following.

Trading TEKs exploiting publicly verifiable lists of *infected* TEKs. As a main contribution we show a smart contract named Take-TEK that allows a buyer (i.e., the adversary willing to spread false positive alerts) to decide the TEKs that will be uploaded by a seller (i.e., the infected individual that is willing to monetize her right to upload TEKs to the servers of the GAEN system). The smart contract requires the buyer to deposit the amount of cryptocurrency (we will call it prize) that he is willing to give to the seller. The seller instead will deposit an amount of cryptocurrency in order to reserve a time slot in which she will try to upload the TEKs. In case she does not manage to complete the upload of the TEKs, the deposit will be assigned to the buyer. The deposit of the seller is therefore useful to make unlikely that a seller tries to prevent other sellers from completing the job.

Take-TEK crucially relies on the server publishing such lists of TEKs along with a signature verifiable with a publicly known public key. We show that the Take-TEK attack can be deployed to generate fake false positive alerts w.r.t. both Immuni and SwissCovid. Indeed, both systems follow strongly the design of GAEN and announce such signed lists of TEKs using ECDSA signatures.

Regardless of Immuni and SwissCovid making available or not their signature public keys, we have successfully extracted the public keys from previously released signatures. Therefore, Take-TEK can be instantiated to attack both (and possibly many more) systems. More details are discussed in Sec. 3.4.2.

Trading TEKs without publicly verifiable signatures: DECO. One might think that realizing the terrorist attack via smart contracts (e.g., Take-TEK) crucially relies on exploiting those signed lists of TEKs under a known (or extractable) public key. At first sight, a fix to such vulnerabilities consists of hiding the public keys and to use a signature scheme such that it is hard to extract the public key from signed messages. However, we show that things are actually more complicated for designers of contact tracing systems. In particular, we show another way to buy/sell TEKs that follows a completely different approach. The key idea is requiring the seller to prove that a TLS session with the server led to a successful upload of the buyer's TEKs. The only requirements on the communication between the smartphone app and the server are that 1) both the TEKs and the positive (or negative) outcome of the upload procedure are part of the exchanged application data in the TLS session, and 2) the upload phase consists of just one request made by the client and the response of the sever (e.g., as it is in SwissCovid). At first sight, the attack seems very hard to realize since notoriously TLS produces deniable communication transcripts when it comes to application data (i.e., exchanged messages are only authenticated and not digitally signed). However, we exploit a recent work of Zhang et al. [126]. They show how to build a fully decentralized TLS oracle, named DECO, for commonly used ciphersuites. Further details are described in Sec. 3.4.3.

Remark on the actual work done by our smart contracts. Both our smart contracts provide full guarantees to both seller and buyer at the expense of running some cryptographic operations that can obviously produce transaction costs. Nevertheless, if we make an additional optimization based on pragmatism, the expensive computations may happen very rarely in practice. Indeed, we notice that the main computational cost for those smart contracts consists of checking at the very end that the seller has completed the task of uploading TEKs correctly. We observe that a buyer can check that TEKs are published by the server on his own. As a result, he would be satisfied in finding out that the trade has been completed successfully. Therefore, it is natural to expect that the buyer would give his approval to the smart contract to transfer the money to the seller avoiding the execution of expensive computations, and therefore saving transaction costs[22]. Since this behavior would be visible in the wild, the reputation of the buyer would also benefit from such approvals and more sellers would want to run contracts with him. Moreover, a (somewhat irrational) buyer that refuses to speed up the execution of the smart contract would anyway not stop the final transfer of the deposited money to the seller. As a result, the buyer would only get a worse reputation. In conclusion, the expensive work done by our smart contracts belongs to pieces of code that would rarely be executed in practice.

3.4.2 Trading TEKs in GAEN Systems

The GAEN API was created to provide an efficient platform for exposure notifications on top of which countries can easily develop digital contact tracing systems. GAEN is supposed to solve various technical problems (e.g., changing BLE MAC address synchronously with the rotation of pseudonyms, keeping BLE advertisements on in background) on a large fraction of available smartphones. At the same time, the API is so inflexible that it forces anyone who is willing to benefit from it to adopt a specific design for contact tracing. What is

[22]Obviously, the smart contract can be adjusted so that, in case the buyer does not give his approval and the seller shows that she completed successfully her part of the contract, the expensive transactions costs due to the lack of help from the buyer are charged to the wallet of the buyer. A simple way to realize this could be asking for an additional deposit made by the buyer. This deposit should clearly cover the transaction costs of the seller in case the buyer does not give his approval, and the seller shows that she successfully completed the upload procedure.

left in the hand of the developers is merely the creation of the graphical interface, the choice of some parameters, the realization of a server to gather and spread data about infected users and, more importantly, an authentication mechanism to avoid the upload of data by non-infected users.

Whenever a user is tested positive, she is given the right to upload her TEKs to the server so that the other users can be notified a risk of infection. The mechanism can be implemented in different ways. For example, a simple method consists of a code generated by the app that is given first to the health operator in order to activate it on the server. Then, once the server has authorized the code, the app will upload the TEKs along with the code (e.g., Immuni follows this approach). More complex mechanisms may be put in place. However, the attack we show next works for every GAEN-based contact tracing system under some natural assumptions that we will discuss later.

In order to evaluate the contagion risk, GAEN provides appropriate methods that take as input two files containing the last TEKs and the related signature. The matching is not performed if the signature does not verify under a public key previously known to Google and Apple. The first file is named `export.bin` and contains, along with other fields, a list of TEKs belonging to infected users that have decided to perform the upload procedure. Each TEK has also a date attached, which indicates when such TEK was used. The second file, named `export.sig`, contains a digital signature of the file `export.bin`[23]. An example of `export.bin` is reported in Fig. 3.13.

3.4.2.1 Take-TEK Smart Contract: Buying/Selling TEKs Uploads

To simplify the description, we will refer to the TEKs file published by the server as a list of pairs of values. In each pair, the first value is a TEK and the second value is the corresponding date of usage `date`.

[23]Apple: Setting Up an Exposure Notification Server https://developer.apple.com/documentation/exposurenotification/setting_up_an_exposure_notification_server. Google: Exposure Key export file format and verification https://developers.google.com/android/exposure-notifications/exposure-key-file-format

```
start_timestamp: 1591254000 //starting time of included keys
end_timestamp: 1591268399 //end time of included keys.
region: "222"  batch_num: 1 batch_size: 1
signature_infos {
verification_key_version: "v1" //version of used verification key
verification_key_id: "222"
signature_algorithm: "1.2.840.10045.4.3.2"
1: "it.ministerodellasalute.immuni"}
keys {
key_data: ".." //base64 encoded TEK
transmission_risk_level: 8
rolling_start_interval_number: 2651616 //date of usage of TEK
rolling_period: 144}...
```

Figure 3.13: An example of an `export.bin` file for Immuni, the Italian contact tracing app. The meaning of the main fields is commented on the side. The `start_timestamp` and `end_timestamp` are expressed in UTC seconds, `rolling_start_interval_number` is expressed in 10 minutes increments from UNIX epoch. The `export.sig` contains the digital signature of the `export.bin` file, along with the field `signature_infos`.

Let the seller P be an infected user who would like to monetize her right to upload TEKs, and buyer B someone who is interested in paying P in order to upload TEKs of his interest. If the seller can prove she acted as promised, this selling process can be executed remotely remaining automated, anonymous, and scalable. GAEN's design requiring the list of TEKs to be signed makes the verification easy to the smart contract, and greatly facilitates such trades. The trade can be performed using a blockchain capable of executing sufficiently powerful smart contracts (e.g., Ethereum). Such smart contract guarantees that P gets an economic compensation if and only if P uploads to the server the TEKs specified by B.

The high-level functioning of the smart contract is as follows. (1) B creates the smart contract posting a list of TEKs with the related date, and deposits a prize to be redeemed by a seller. (2) An interested P makes a small deposit to declare her intention to upload the TEKs specified by B (the purpose of this small deposit is explained later). After having made this deposit, (3) P has a specified amount of time to complete the upload procedure. Before the time runs out, P must provide a list of TEKs which includes all the pairs $(\mathsf{tek}, \mathsf{date})$ specified by B, along with a valid signature under the server's public key. If P manages to do so, she gets a reward, otherwise both deposits go back to B.

By making a deposit, the seller reserves a time slot during which she can perform the upload. Such deposit protects the buyer from denial of service (DoS) attacks by sellers who actually do not have the right to upload TEKs. Here, as in the remainder of the paper, with the word DoS we mean attacks carried out by fake sellers which prevent honest sellers from participating to the trade.

We name the above smart contract Take-TEK and the attack that leverages the use of this smart contract Take-TEK attack. The time window given to P must be wide enough to take into account that new TEKs are not continuously released by the server, in fact, several hours may pass between the submission of a TEK and its publication. Obviously, the amounts of both deposits will be significantly higher than transaction fees. A custom software is needed to upload arbitrary TEKs to the server. However, this simple software may be developed even by other entities (not just the buyers), and publicly distributed on the Internet or other sources (e.g., Darknet). Therefore, all the seller would need to do is just running a software on a smartphone/-computer; something that is easily doable by a large fraction of the infected citizens willing to gain money[24]. Additionally, the time given to the seller to complete the upload after having been tested positive must be long enough to reserve a slot on the blockchain (i.e., enough to wait that the transaction related to the seller's deposit gets confirmed) and subsequently send the TEKs via the custom software.

Attack description. B and P own wallets pk_B and pk_P respectively. The buyer has no assurance that the seller is actually an infected person, and she is not just a malicious party trying to slow down the

[24]COVID-19 by itself caused a global economic crisis which led to lower wages and job losses. More details at https://en.wikipedia.org/wiki/COVID-19_recession.

buyer's plan. Thus, some collateral must be deposited by P too. The seller will lose the collateral deposit in case she is not able to prove that she sent the buyer's TEKs to the server S. Let the signature scheme used by GAEN be $(\mathsf{Gen}_\mathsf{S}, \mathsf{Sign}_\mathsf{S}, \mathsf{Verify}_\mathsf{S})$. The protocol description is depicted in Fig. 3.14 and a brief overview of the main functions follows below.

`Constructor`$(\mathbf{T}_\mathsf{B}, \mathsf{vk}_\mathsf{S}, t, d_\mathsf{P})$**:** It takes as input a set of tuples $\mathbf{T}_\mathsf{B} := (\mathsf{tek}_i^\mathsf{B}, \mathsf{date}_i^\mathsf{B})_{i\in[n]}$ with $n \leq \mathsf{maxteks}$ [25], where tek_i is the i-th TEK of the buyer and date_i is the associated date, the verification key vk_S to be used to verify the signature of the TEKs list, a timestamp t, indicating the maximum time the seller has to provide the correct list and signature, and the collateral value d_P that the seller must deposit.

`Deposit`()**:** must be triggered by B and takes as input a quantity p of coins as the payment for the seller.

`Promise`()**:** must be triggered by the seller P by sending a quantity of collateral deposit d_P as a payment when invoked.

`SendTeks`$(\mathbf{T}_\mathsf{KS}, \sigma_T)$**:** must be triggered by the seller P to provide a list of TEKs together with its signature σ_T. Let the list released by the server be $\mathbf{T} = (\mathsf{tek}_i, \mathsf{date}_i)_{i\in[N]}$, where N is the number of published TEKs. It checks that:

- $\mathbf{T}_\mathsf{B} \subseteq \mathbf{T}$ and $\mathsf{Verify}_\mathsf{S}(\mathbf{T}, \sigma_\mathbf{T}; \mathsf{vk}_\mathsf{S}) = 1$.

If the checks passes, d_B coins are transferred to the seller's wallet pk_P.

3.4.2.2 On the Practicality of Take-TEK Attack

Various proposed upload authorization mechanisms include manual steps (e.g., SwissCovid uses an authorization code, termed covidcode, which lasts for 24 hours) which, in order to function properly, naturally

[25]The maximum number of TEKs that can be uploaded in one shot depends on the particular ACT.

Take-TEK Attack

We consider two entities: the seller P and the buyer B, with wallets pk_B and pk_P respectively. The protocol works as follows:

1. B invokes the constructor, taking as input the buyer's TEKs list $\mathbf{T}_\mathsf{B}$, the server verification key vk_S that will be used to verify the signed TEKs list, a timestamp t, and a value d_P indicating the minimal amount that P must deposit in order to participate.
   After having created the contract, B triggers the function `Deposit` to deposit the prize p aimed for the seller who uploads $\mathbf{T}_\mathsf{B}$ to the server.

2. P deposits her collateral by triggering the function `Promise`. Now the seller has at most time t to send a TEKs list $\mathbf{T}$ signed by the server.

3. If P, before time t, triggers the function `SendTeks` submitting a signed TEKs list $\mathbf{T}$ such that it satisfies conditions $\mathbf{T}_\mathsf{B} \subseteq \mathbf{T}$ and $\mathsf{Verify}_\mathsf{S}(\mathbf{T}, \sigma_\mathbf{T}; \mathsf{vk}_\mathsf{S}) = 1$, the collateral deposit d_P of P and the prize p are transferred to P's wallet. Otherwise, if time t has passed, they are moved to B's wallet.

Figure 3.14: The steps followed by buyer B and seller P to carry out the Take-TEK attack.

give the seller enough time to perform the steps mentioned in the section above. For example, if a code is communicated to the infected user via a phone call, she should be given a fairly large amount of time to write down the code and insert it in the app later on (the needs of people with disabilities and of elder people must be taken into account). Even systems that have fairly strict requirements on the time by which the upload procedure must be completed should allow for errors and recovery procedures, which may give additional time to the future seller. For example, Immuni requires that the infected user dictates, via phone call, a code that appears on her device. After that, the user must complete the upload within two minutes. If this does not happen, the procedure must be repeated. Additionally, the system should be tolerant. People should have the opportunity to perform the upload procedure later on if they are unable to do it in that precise moment. It is worth noticing that strict requirements on the upload phase reduce users' privacy. A clear example is Immuni, where the medical operator, by checking whether a code has been used or is instead expired, gets to know whether or not the infected user actually uploaded her TEKs.

Implementation. We implemented Take-TEK as a smart contract for Ethereum, published it in a public repository[26], and tested it locally. Since Ethereum does not use `ECDSA-SHA256` (i.e., the one used in GAEN) for built-in transaction signature verification, there is the need to use specific solidity smart contract libraries[27] which lead to extra gas usage. Considering the exchange rate of 206 dollars per ETH token on the 20th of July 2020, signature verification costs around 11 dollars (1235000 of gas). In order to compute the full cost, one should add about 0.4 dollars (45000 of gas) for each TEK that is contained in the export.bin file[28]. Notice that in principle our smart contract can handle export files large as the maximum data that an Ethereum transaction can handle (i.e., around 44 Kbytes). However, by splitting the export files in multiple chunks to be sent to the smart contract in different transactions, one can easily overcome such limitation. Furthermore, to avoid the expensive (in terms of gas) operation of storing such chunks on the smart contract, one can make the following optimization. Instead of storing the chunk, the smart contract will just store the intermediate hash values according to the Merkle-Damgård architecture of SHA-256. Such value is computed on input both the current chunk and the intermediate hash value produced from the pre-

[26]Code available at https://github.com/danielefriolo/TEnK-U.

[27]The library we used for signature verification is available at https://github.com/tdrerup/elliptic-curve-solidity.

[28]The cost of 45000 of gas includes TEKs extraction from the file as well as checking if the stored TEKs are in the extracted ones. To simplify the gas evaluation, we assume that B stores only one TEK in the contract.

vious chunk. When processing a chunk, the smart contract inspects it and marks the buyers' TEKs contained in the chunk. The smart contract also takes care of possible splits of a TEK between two chunks. When the last chunk is submitted, the smart contract verifies that all the buyers' TEKs were included. Then, it verifies the signature according to the final hash value computed using the accumulated intermediate hashes.

3.4.2.3 Subtleties in the Wild

In Sec. 3.4.2.1, we gave a high-level overview of how TEKs uploads can be sold safely via blockchains. However, there are some subtleties we overlooked for the sake of simplicity. We first analyze the advantages for adversaries when using automated trade compared to already known attacks. Then, we consider certain problems that arise while trying to concretely mount our attack against deployed GAEN-based contact tracing systems. We also show how these difficulties are easily tackled if very small modifications to our attack are made.

Advantages of automated trade (for an adversary). One might think that malicious injection of fake TEKs is inherent in decentralized contact tracing systems since there is no control over the smartphone used by infected individuals and thus, when the time of the upload comes, the infected person can always use a smartphone belonging to someone else.

While it is true that such simple attacks are very hard to tackle, they have limited impact for at least two reasons: 1) the buyer must handover his smartphone to the seller, and this requires physical proximity; 2) sellers and buyers must trust each other since an illegal payment must be performed without being able to rely on justice in case of missing payment or aborted upload of keys. Indeed, even if in need of money, people are generally afraid of dealing with criminals since they may get scammed or threatened. Additionally, the buyer might expose the sellers' identities to the authorities in case he gets legally persecuted. Equally, the buyer may share the same concern w.r.t. an unreliable seller. It goes without saying that some citizens are prone

to violate the rules[29] when they believe that risks are low compared to the advantages.
As such, attacks involving the exchange of smartphones, or the usage of a malicious app uploading TEKs sent by a criminal contacted directly by the infected citizen, do not scale and their damage may be considered tolerable. Having a mechanism which allows this trade to happen remotely, in anonymity, and ensuring no party is cheated, solves all the above problems for parties willing to abuse contact tracing systems. Indeed, it provides a framework for large-scale black markets of TEKs. The seller would not feel threatened in any way and could easily earn money, on the other hand, the buyers would benefit from a larger set of users to be in business with, therefore succeeding in many possible attack scenarios. Other systems for black markets based on reputations could also be used, but they are clearly less appealing than the transparency and usability of mediator-free smart contracts.

ACTs and data minimization. The effectiveness of ACTs is strictly related to various factors among which the percentage of active population using them. Appropriate measures should be taken to earn citizens' trust since it is the only way to guarantee broad adoption. With this in mind, the European Commission released a series of recommendations in relation to data protection stating the need of identifying solutions that are the least intrusive and comply with the principle of data minimization [56]. A similar recommendation has been given by the Chaos Computer Club (CCC)[30], the Europe's largest ethical hackers association, which explicitly states that "data which is no longer needed must be deleted". Corona-Warn, the German contact-tracing system, declares to be fully compliant with CCC's guidelines[31]. Many other systems are inspired by similar principles. For example, the Ital-

[29]The infected person also commits a violation by allowing the injection of fake TEKs.

[30]10 requirements for the evaluation of "Contact Tracing" apps `https://www.ccc.de/en/updates/2020/contact-tracing-requirements`. Accessed August 2020.

[31]Criteria for the Evaluation of Contact Tracing Apps `https://github.com/corona-warn-app/cwa-documentation/blob/ec703906c109bd7c3cc84bc361b7e703b20650ea/pruefsteine.md`. Accessed August 2020.

ian system Immuni also declares that data is deleted when no longer needed[32], as well as the Swiss system SwissCovid which also specifies a retention period for the TEKs and the upload authorization codes [33]. In its recommendation to build a verification server authenticating the uploaded TEKs, Google states that identifiable information should not be associated with uploaded data[34].
The adoption of the above measures ensures that uploaded data do not link to, nor identify a particular individual. This is very important considering that GAEN systems are vulnerable to the paparazzi attack.

Evaluation of seller's risks. Considering the above data minimization principles, are the seller and the buyer at risk of being legally persecuted for a trade that may be deemed as illegal? The answer seems to be no. If data is handled as specified above, there would be no way to associate the seller to her uploaded TEKs at a later time. Data exchanged during the attack would also not directly compromise neither the buyer nor the seller[35].

However, there is a problem for a seller who really wants to minimize the chance of getting caught. In fact, since the TEKs proposed by the buyer are posted in clear on the blockchain, authorities may become aware of them and activate ad-hoc procedures monitoring the incriminated TEKs and exploiting the upload authorization process to identify the guilty seller. This does not seem to directly contradict the data minimization principle when national security is at stake. If

[32]https://github.com/immuni-app/immuni-documentation.

[33]https://www.bag.admin.ch/bag/en/home/krankheiten/ausbrueche-epidemien-pandemien/aktuelle-ausbrueche-epidemien/novel-cov/swisscovid-app-und-contact-tracing.html. Accessed August 2020.

[34]Google: Exposure Notification Verification Server https://developers.google.com/android/exposure-notifications/verification-system. Accessed August 2020.

[35]In this analysis, we refer only to contact tracing system data and messages exchanged via the blockchain during the execution of the attack. We do not take into account border-line situations as, for example, the case where there is only a single infected individual. We also ignore additional information that may help investigators figuring out who the seller is, for example how the money are spent after the trade.

the server getting the TEKs upload monitors the requests (e.g., by storing connection logs) without colluding with the health authority, the seller could be easily incriminated after the TEKs have been detected in the smart contract by just looking at her IP stored together with such request. However, in this case, the usage of an anonymity service like Tor [52] can easily reduce the chance of getting caught. If the authorities are colluding, the upload authorization codes (e.g., the covidcode) may be associated with the identities of infected users, and TEKs could be in turn mapped to a precise individual via such codes. However, by slightly increasing the complexity of the smart contract, such risk may be completely avoided. It suffices for the buyer to encrypt his TEKs with a public key provided by the seller, who then will use a non-interactive zero-knowledge (NIZK) proof system to prove that the TEKs encrypted under the specified public key are indeed contained in the list signed with the server's public key. This requires an additional interaction with the buyer, who has to publish the encrypted TEKs. Once again, the seller is protected by a timer which assigns her all the deposits if the buyer does not reply. Efficient Ethereum implementations of NIZK proofs are known in literature, like NIZKs for Σ-protocols [120] or zk-SNARKs [103, 130, 131].

Even if the buyer decides to claim the authorship of the attack at a later point in time (e.g., as it usually happens for terrorist attacks) by opening the encrypted values on the blockchain to published TEKs, the seller would not be at risk if data was handled according to the principles of data economy and anonymity. Any evidence based on contact tracing data would be a clear indicator that those principles have been violated. This could result in a big disincentive in using the app, since citizens may think (probably rightfully) that data could also be abused for other reasons, perhaps for mass surveillance purposes. Finally, we want to point out that even if several researchers raised the concern about the possible birth of black markets [99, 116], we did not find any document related to any ACT, either issued by governments or national security agencies, which deeply evaluates these risks. To the best of our knowledge, no risk analysis ever mentions to monitor the dark web and blockchains looking for suspicious smart contracts. It goes by itself that if blockchains are not monitored, all extra measures taken in this paragraph to protect the seller are not necessary.

Other subtleties. There are two other subtleties with limited impact to consider for the actual realization of the attack. We describe them in Sec. 3.4.4.1 and Sec. 3.4.4.2 and shortly mention them here:

- *Extracting public keys from signatures*: Generally, servers' public keys do not seem to have been made publicly available neither by Google and Apple, nor by the countries which deployed GAEN-based ACTs[36]. However, the employed signature algorithm (i.e., ECDSA) allows to retrieve this public key starting from a pair of signed messages.

- *Updates of public keys*: The structure of the `export.bin` file allows for updates of the used digital signature key (see Fig. 3.13). Therefore, it might happen that, after the seller makes the deposit and accepts to upload the buyer's TEK, the server, by coincidence, decides to use a new key which was never used before. Thus, the server would produce a signature that is not verifiable under the public key posted on the smart contract. However, by making a slight modification to the smart contract, it is possible to handle also this unfortunate event. Moreover, keys have changed very rarely in export files.

3.4.3 Connecting Smart Contracts to TLS Sessions

The Take-TEK attack relies on the fact that a digital signature is used to authorize uploads. Additionally, the ability to extract the public key from signed messages may also play a key role. Therefore, one might think that to protect GAEN systems the public key should remain hidden and the signature scheme should be such that extracting the public key from message-signature pairs is hard. In this way, due to the inability of the smart contract to verify that a TEK is officially in a list of infected TEKs, the attack would fail. However, things are not

[36]Once an ACT hands its public key over to Google and Apple, it can completely rely on GAEN APIs to perform signature verification without storing the public key in clear in the app source code (see https://developers.google.com/android/exposure-notifications/exposure-key-file-format for more details).

so easy. Indeed, we show that TLS oracles can be used to prove to a smart contract that an upload was successfully performed, without relying on signatures of TEKs.

3.4.3.1 Decentralized Oracles

Recently, Zhang et al. [126], introduced the concept of Decentralized Oracles. Roughly, an oracle is an entity that can be queried by a client to interact with a TLS server and help the client proving statements about the connection transcript. Previously known oracle constructions rely on trusted/semi-trusted execution environments [125], thus not giving any help in our case. DECO [126] is the first work where a fully-decentralized construction is proposed for specific ciphersuites such as CBC-HMAC and AES-GCM coupled with DH key exchange with ephemeral secrets. We recall that a TLS connection is divided in two parts: a handshake phase where key exchange is performed, and a phase during which the transferred data is encrypted/decrypted by the client/server using the key exchanged in the previous phase. GAEN servers usually accept Elliptic-Curve Diffie-Hellman key Exchange (ECDHE) for the first phase, while for the second phase some servers accept only AES-GCM (e.g., Immuni), whereas others, like SwissCovid's one, accept also CBC-HMAC as a ciphersuite. To guarantee the integrity of data, the plaintext is usually compressed and a MAC on the compressed string is calculated using a key derived from the DH exchanged key.

Decentralized Key-Exchange. We provide below an informal description of the key-exchange in DECO for ECDHE that is called Three Party Handshake (3PHS).

We assume three entities: a prover P, a verifier V, and a server S. P and V jointly act as a TLS client. The overall idea of DECO is that the prover and verifier, after performing some two-party computations, compute shares of the exchanged key, while the server computes the entire key without even noticing that P and V are two distinct interacting entities.

When using CBC-HMAC, the keys $k_{\mathsf{P}}^{\mathsf{MAC}}$, $k_{\mathsf{V}}^{\mathsf{MAC}}$ (such that $k_{\mathsf{P}}^{\mathsf{MAC}} + k_{\mathsf{V}}^{\mathsf{MAC}} = k^{\mathsf{MAC}}$) are learned by P and V respectively, while k^{Enc} is only known to P. When using AES-GCM, the same key is used for both encryption and MAC, therefore both P and V just get a share of it. While P and V only learn their secret shares of the key, the server S gets to know both k^{Enc} and k^{MAC}.

Let G be an EC group generator. The key exchange phase works as follows:

- P establishes a TLS connection with the server S.
- When receiving the DH share $Y_{\mathsf{S}} = s_{\mathsf{S}} \cdot G$ from S, P forwards it to V[37].
- V samples a DH secret s_{V} and sends his DH share $Y_{\mathsf{V}} = s_{\mathsf{V}} \cdot G$ to P.
- P samples her DH secret s_{P}, calculates her DH share $Y_{\mathsf{P}} = s_{\mathsf{P}} \cdot G$, calculates the combined DH share $Y = Y_{\mathsf{P}} + Y_{\mathsf{V}}$, and sends Y to S.

Finally, S computes the DH exchanged key as $Z = s_{\mathsf{S}} \cdot Y$. P and V will compute their secret shares of Z as $Z_{\mathsf{P}} = s_{\mathsf{P}} \cdot Y_{\mathsf{S}}$ and $Z_{\mathsf{V}} = s_{\mathsf{V}} \cdot Y_{\mathsf{S}}$. Note that $Z_{\mathsf{P}} + Z_{\mathsf{S}} = Z$, where $+$ is the EC group operation. Now that P and V have secret shares of EC points, they use secure two-party computation (2PC) to evaluate a PRF (that we call TLS-PRF) to derive the keys $k_{\mathsf{P}}^{\mathsf{MAC}}$ and $k_{\mathsf{V}}^{\mathsf{MAC}}$. The authors face and solve several challenges in order to derive keys efficiently via 2PC. We do not cover this part, a more detailed description can be found in [126].

Encrypted communication. At the end of the 3PHS, P and V have to engage in a 2PC protocol to correctly calculate the MAC and the encryption on the plaintext to be sent to the server, without revealing the shares to each other. Privacy of the plaintext is also ensured with respect to V. For CBC-HMAC, the encryption of such plaintext is computed exclusively by P who holds the encryption key. The authors [126] provide hand-optimized protocols which are much

[37] P would also receive from S a certificate and a digital signature that certify the server's identity. P would also forward them to V to convince him that an interaction with the desired server is taking place.

more efficient then the ones obtained by directly applying 2PC techniques. The 2PC protocol for AES-GCM is a lot slower than the one for CBC-HMAC since for AES-GCM P and V must cooperate also for the encryption.

Proving statements. An important feature of DECO is that P, when the communication with S ends, can prove, in zero knowledge, statements on the communication transcript in a very efficient way. However, to make their protocol practical for our goal, we do not try to maintain the transcript private. As a result, we will not discuss this part of DECO which can be found in [126]. In the following, we describe how to adapt DECO to our scenario.

3.4.3.2 A Smart Contract Oracle

Our goal is to make the smart contract play the role of the DECO verifier. In this way, the smart contract would be able to verify that the intended communication between the seller and the server took place and to reward the seller accordingly. Unfortunately, we can not just plug DECO into a smart contract for several reasons. For example, DECO requires to run intensive 2PC related tasks, to sample random values, and to maintain a private state[38].

Therefore, we keep running the DECO protocol off-chain but we find a way to connect the DECO run between the prover and the verifier to the state of the smart contract, so that the smart contract will eventually be able to act as an impartial judge punishing the malicious party when a deviation from the prescribed honest behavior is detected. In particular, the seller acts as a prover and the buyer as a verifier, and we guarantee no party is able to cheat (i.e., the seller is paid if and only if she performs the upload of the requested TEKs) by binding the off-chain execution to the state of the smart contract itself. Furthermore, we guarantee privacy of the messages exchanged between the server and the prover only until their TLS connection is open. After the communication ends, the seller proves that she acted honestly by providing the application-level messages exchanged with the server,

[38]Keeping a private state inside a smart contract is not possible and computationally intensive operations generate high costs.

along with the corresponding MAC tags w.r.t. the MAC key which is bound to the smart contract. To be more specific, the smart contract freezes a share of the MAC key and the seller has to show a communication transcript (i.e., the messages exchanged with the server and corresponding MAC tags) which is consistent with such share. Privacy of the upload request message to be sent to the server is crucial while the TLS session is open because the verifier may abort the protocol and use the authorization token of the prover to upload data by himself without paying out the promised reward. On the other hand, making the communication public after it took place does not endanger the prover, apart from the considerations made in Section 3.4.2.3, and makes the verification procedure much more practical. What we need is that the shares of the prover and the verifier are kept private until the end of the protocol, and then revealed to the smart contract, along with other information, for verification and reward paying. In addition, the TLS session timeout should be big enough to allow for the 2PC execution. To this regard, Zhang et. al already verified the practical feasibility of their protocol [126]. Obviously, P must know how to reach V to carry out the protocol. To address concerns regarding anonymity, V may set up a Tor hidden service[39]. Using hidden services may significantly slow down the process, however we found both Immuni and SwissCovid servers to give a generous time out window of two hours[40]. Another point to consider is that upload authorization tokens may have a limited duration. For example, in SwissCovid, the smartphone, by interacting with an appropriate server (different from the TEKs upload server, called CovidCode-Service), exchanges the covid code for a signed JWT token that is valid for 5 minutes[41]. Then, this token is sent by the smartphone to the server along with the TEKs to complete the upload. Thus, the upload message containing the TEKs and the authorization token must be computed and

[39]More on Tor hidden services can be found at `https://2019.www.torproject.org/docs/onion-services`.

[40]Interestingly, in June 2020 the timeout of a TLS session with both Immuni and SwissCovid upload servers was limited to 5 minutes, but it has been then extended to two hours.

[41]See CovidCode-Service configuration `https://github.com/admin-ch/CovidCode-Service/blob/develop/src/main/resources/application-prod.yml`.

sent to the server within 5 minutes from the reception of the JWT token. Given the high efficiency of DECO when CBC-HMAC is used, even when bandwidth is limited [88], it is reasonable to think that the attack is feasible in SwissCovid. In Immuni instead, no signed token is used. In fact, the upload must be completed within 2 minutes after the infected user has communicated the code to the health operator. Therefore, in Immuni the attack would less likely be operative, especially with Tor, given that the slower AES-GCM ciphersuite is required.

Protocol description. From now on, we refer to the seller and the buyer as prover P and verifier V respectively; we denote the server as S. In the following, we explain the detailed attack for the CBC-HMAC ciphersuite. When creating the smart contract, aside depositing a prize for P, V also posts the DH share $Y_{\mathsf{V}} = s_{\mathsf{V}} \cdot G$ he is willing to use during the 3PHS, along with requested TEKs (and dates).

First, P transacts on the smart contract to reserve a time slot of duration t_1 by which a DECO protocol run must be performed together with V and S, and the data needed to redeem the reward must be posted on the smart contract by P. If time t_1 elapses, P loses her slot. This reservation mechanism is needed to prevent V from getting back the reward while an honest P performs the upload of the requested TEKs. In fact, the verifier could also act as a prover and simulate a reward-paying interaction with the server to the smart contract, which would have no mean to distinguish it from a fake one[42]. By adding a reservation mechanism, we are sure a malicious V cannot play a simulated transcript in the smart contract while honest P is performing with him the DECO protocol run. Furthermore, since the communication for the upload between the server and the prover consists of just a single query followed by a single response, it is not possible for a cheating verifier to make the timer expire avoiding to pay the prover while at the same time the upload of the TEKs successfully completes. In fact, once V cooperates with P to build a valid request,

[42]During the 3PHS handshake V will also forward to P a digital signature from S. This signature convinces P that V is interacting with the desired S. However, our smart contract ignores this detail.

S will reply to P independently of what V does, thus giving P all she needs to redeem the reward.

When executing the 3PHS, P checks that the value Y'_{V} sent by V during the handshake corresponds to the value Y_{V} posted on the smart contract. This prevents V from providing an erroneous DH share and blaming P for it. If this is not the case, P aborts. Since no upload message has been sent to the server yet, no party gains advantage from this operation. If V's share is correct (i.e., $Y_{\mathsf{V}} = Y'_{\mathsf{V}}$), parties engage in the communication with S and jointly compute the MAC (via 2PC as in [126]) on the upload request m_c generated by P . If the connection ends successfully[43], the elected P posts (only who reserved this slot is allowed to post this message) on the smart contract the following:

- The entire communication transcript, that is (m_c, m_s) together with the MACs (θ_c, θ_s), calculated by the client(s) P $\leftrightarrow$ V and the server S.
- The prover's secret s_{P}.
- The DH share of the server Y_{S} received during the 3PHS.

Then, the smart contract starts a timer t_2 indicating the maximum time V has to reveal his secret s_{V}. In case V does not do that, the prize is automatically transferred to the seller P. If V reveals s_{V}, the smart contract does the following:

- Check that $Y_{\mathsf{V}} = s_{\mathsf{V}} \cdot G$ and if not, transfer the prize to P.
- If the check passes, reconstruct the secret Z from $s_{\mathsf{V}}, s_{\mathsf{P}}, Y_s$, and apply TLS-PRF to derive the MAC key k^{MAC}.

Now the smart contract has everything it needs to check that the fields inside message m_c (from the prover to the server) are correct (i.e., the buyer's TEK are present), the response message (from the server to

[43]This can be inferred from the communication. For example, as in SwissCovid (see SwissCovid Server Controller: https://github.com/DP-3T/dp3t-sdk-backend/blob/a730a5b276591e5cc8b6c609e2b0ba29c6069eb6/dpppt-backend-sdk/dpppt-backend-sdk-ws/src/main/java/org/dpppt/backend/sdk/ws/controller/GaenController.java), S may reply P with either a success message such as “200 OK” or an error message.

the prover) is positive, and that the MACs (θ_c, θ_s) verify w.r.t. k^{MAC}. If all the checks pass, the prize is transferred to P, otherwise P gains no prize and the deposit is returned back to V.

As mentioned before, V is not encouraged to provide a different public key w.r.t. the one he used in DECO execution, otherwise P will just abort. On the other hand, the prover is not able to earn a reward without uploading the promised TEKs. In fact, the probability for the prover to come up with a pair (m'_c, θ'_c) (resp. (m'_s, θ'_s)) that verifies under the key k'^{MAC} derived from $Z' = Z'_{\mathsf{P}} + Z'_{\mathsf{V}}$ with $Z'_{\mathsf{P}} := s'_{\mathsf{P}} \cdot Y'_{\mathsf{S}}$ and $Z_{\mathsf{V}} := s_{\mathsf{V}} \cdot Y'_{\mathsf{S}}$ is negligible due to the fact that s_{P} is fixed and honestly generated, thus randomizing Z', hence k'^{MAC}.

A note on DoS attacks. It is important to prevent DoS attacks run by sellers who actually do not have the right to upload TEKs and end up by just wasting the buyer's precious time. In the previous discussion this protection is not provided: before sending the jointly computed message (m_c, θ_c), the seller can decide to not forward the message to the server. Now, the buyer has to open his commitment to show his secret s_{V} in order to not lose the prize. As a result, the committed value cannot be used in other runs. To address this issue, the smart contract can be modified to handle multiple sessions. Instead of storing Y_{V} as a single DH contribute, the buyer stores the root of a Merkle tree. Now, when the seller interacts with the contract to reserve a session, a session id (a simple counter j suffices) is assigned to her: the DH contribute used in the 3HPS will correspond now to the j-th leaf of the Merkle tree. Now, when the buyer has to open his secret s_{V}, he also reveals the path of the Merkle tree from the root to the leaf j. The smart contract will now verify that the contribute is correctly derived from the root by following a path with correct openings. Let us consider a Merkle root committing to 2^k elements, thus allowing the buyer to open as many sessions. For a k large enough, a malicious seller should spend a considerable amount of money in order to reserve all the sessions.

3.4.4 Other Subtleties: Details

In this section, we report more details on some subtleties about our attacks we skimmed on in the previous sections for the sake of clarity.

3.4.4.1 Extracting Public Keys from Signatures

Take-TEK (cfr., Section 3.4.2.1) requires that the server's public key is known to both the involved parties. This guarantees that the buyer is sure the reward is paid only to sellers who actually upload data to the ACT, and that honest sellers are sure they will be able to satisfy the conditions to be paid, namely obtaining a valid digital signature for reward redemption. A GitHub issue asking for the public key of the Italian contact tracing app was opened on the 7th of June 2020 and it has still not been addressed at the time of writing[44]. The SwissCovid Android app contains a configuration file specifying the production version of the bucket public key[45] that is used to perform signature verification outside GAEN. Anyway, as we can notice with Immuni, this is not a requirement. One might think that keeping the verification keys secret may prevent attacks as the one of Section 3.4.2.1. However, it turns out that it is actually not the case. In fact, since GAEN uses ECDSA, starting from a signature and the related message we can recover two candidate public keys, one of which will match the actual one with overwhelming probability. A practical example showing this procedure can be found in [121]. Such message/signature pairs are generally made publicly available and are easily accessible by appropriately querying the server of the specific contact tracing system. Multiple pairs per day may be released. A comprehensive description on how to get this data has been provided by the Testing Apps for COVID-19 Tracing (TACT) project, along with scripts to automate the downloading process [83]. We also practically performed the extraction procedure, successfully extracting the keys for both SwissCovid and Immuni.

[44]See https://github.com/immuni-app/immuni-documentation/issues/114.

[45]The value BUCKET_PUBLIC_KEY can be found in https://github.com/DP-3T/dp3t-app-android-ch/blob/master/app/backend_certs.gradle.

3.4.4.2 Updates of Public Keys

There is a subtle technical problem with the attack described in Section 3.4.2.1. The digital signature keys that the server uses may change over time. In fact, as shown in Fig. 3.13, the `export.bin` file includes a field indicating a version for the verification key. This field follows a progressive numeration, that is the first version is termed v1, the second one v2 and so on. This means that the server may change the verification key it uses, perhaps within a set of keys that have been pre-shared with Google and Apple. Therefore, it might happen that, after the seller makes the deposit and accepts to upload the buyer's TEK, the server, by coincidence, decides to use a new key which was never used before, thus producing a signature that is not verifiable under the public key posted on the smart contract.

However, by making a slight modification to the smart contract, it is possible to handle also this unfortunate event. Having realized that she would be unable to redeem the reward, the seller might activate a special recovery condition. After this, the buyer will be able to collect both deposits if and only if he manages to provide a pair of export files which have an `end_timestamp` (Fig. 3.13) subsequent to the time of the recovery request and verify under the public key originally posted on the smart contract; otherwise the deposits are returned to the original owners. Obviously, enough time should be given to the buyer to provide the export files, similarly to what happens to the seller after her deposit.

This event is certainly very annoying for the seller and might play as disincentive to join the trade, but taking a look at real-world data one realizes that this is a relatively rare event. We considered several countries which are currently using a digital contact tracing system, namely: Italy, Switzerland, Austria, Germany, Ireland, Northern Ireland, Denmark, Latvia, Canada and US Virginia. Until January 13th 2021 (last time we checked), only US Virginia and Italy have switched to the second version of the verification key. In particular, the change to the Italian system dates back to the 15th of June 2020[46] and no modifications have been made since then. To the best of our knowledge, the criteria by which the verification key should change is not

[46]This change occurred in the 4th export file.

documented anywhere.

3.4.4.3 CBC-HMAC vs AES-GCM

Differently from CBC-HMAC, AES-GCM relies on the same key for both encryption and MACs. The impact of AES-GCM is twofold: 1) more computation is needed to perform the required 2PC to calculate messages from/to the server, due to the AES algorithm itself, 2) the prover does not learn the encryption key after 3PHS, meaning that both encryption and decryption must be done via 2PC as well. On the smart contract side, this difference boils down to a lack of fairness. After V and P have calculated together the upload message and sent it then to S, V could decide not to help the prover to decrypt the server's response. Now, P has no witness in her hands to give to the smart contract in order to prove that she has correctly performed the TEKs upload. As a result, she cannot redeem the prize. The problem can be easily solved by giving to the smart contract the burden of decrypting the server's ciphertext. In our approach, V must commit to his key and open it later. When this happens, the smart contract reconstructs the MAC/encryption key, decrypts the ciphertext, does the necessary checks, and pays the prize to P. The CBC-HMAC version of DECO is way faster then the AES-GCM one. However, looking at practical evaluations made by the authors [88, 126] it is reasonable to think that all their solutions may fit in the time window given by contact tracing servers (e.g., 2 hours in Immuni and SwissCovid) for the TLS connection[47], even when hiding V through Tor hidden services. What is less likely is that, in the case of Immuni which uses AES-GCM and requires the upload to be completed within two minutes, the upload request message (m_c, θ_c) is computed and sent to the server in time; especially when the prover and the verifier communicate via Tor.

[47]Full TLS connection logs dating to June 2020 can be found in [15].

Chapter 4

Threshold Cryptography

In this chapter, we provide new contributions in the area of threshold cryptography. We first consider proofs over threshold relations (cfr., Sec. 4.1). In this setting, a statement is composed of ℓ instances and a prover wants to prove, in zero knowledge, that it knows a witness for at least k of them. We provide a generic construction that starting from the Σ-protocols for proving knowledge of a witness for the individual instances outputs a Σ-protocol for the threshold relation (cfr., Sec. 4.1.4). The input Σ-protocols have to be stackable Σ-protocols (cfr., Sec. 4.1.4.1) and the output protocol is still a stackable Σ-protocol. Our result improves prior results either in terms of communication complexity, or of generality (cfr., Tab. 4.1). We then show how to distribute the computation of the prover and use this to obtain a threshold ring signature (cfr., Sec. 4.1.5). Our threshold ring signature has interesting features such as reduced interaction among the signers, and support for hierarchical access structures.

Then, we build a new extendable threshold ring signature (ETRS) with enhanced anonymity properties (cfr., Sec. 4.2). Our ETRS is based on a novel tool we formalize and construct called extendable non-interactive witness indistinguishable proof of knowledge (ENIWI PoK) (cfr., Sec. 4.2.7 and Sec. 4.2.10). An ENIWI PoK is a NIWI PoK for a threshold relation that can be non-interactively updated by many different provers to increase either k or ℓ. We then show how to construct an ETRS starting from an ENIWI PoK and other standard cryptographic tools. We improve previous constructions in terms of anonymity guarantees, size, and time complexity (cfr., Tab. 4.2 and Tab. 4.3). The strong anonymity guarantees of our ETRS allow to run public anonymous petitions where the supporters (who desire to remain anonymous) do not have to trust each other.

While confronting ourselves with previous works [5, 64] we found examples of pitfalls of the provable security paradigm such as fallacies in the security proofs and inadequate definitions. We analyse and provide solutions to such issues. For example, we show that the proof of EHVZK in [64] was incorrect. To this regard, we provide new definitions of 1-out-of-2 equivocal commitment scheme as well as revised security proofs (cfr., Sec. 4.1.6). Regarding ETRS we show that the anonymity definition proposed in [5] is inadequate for real-world count-me-in applications. We show that such inadequacy concretely shows up in the most efficient ETRS proposed in [5] (cfr., Sec. 4.2.3). To fix this issue, we propose stronger definitions as well as an ETRS satisfying such definitions.

Parts of the results of Sec. 4.1 were published in [11]. The results of Sec. 4.2 will appear in the proceedings of PKC 2023. In the meanwhile, we published such results on ePrint [10].

4.1 Proofs over Threshold Relations

With the advent of blockchain technology and cryptocurrencies, there has been more interest in designing practical systems for decentralized computations. In particular, there is an effort towards systems producing succinct messages that can later be uploaded on blockchains guaranteeing some public verifiability. Notable examples of such tools are threshold signatures and succinct non-interactive arguments of knowledge (SNARKs). W.r.t. the above motivation we focus on proofs over threshold relations (PTRs) where a statement consists of ℓ instances and the prover wants to prove knowledge of witnesses for at least k of them. For simplicity, we will refer to such a proof as a (k, ℓ)-PTR.

4.1.1 Related work

Several previous works have focused on obtaining (k, ℓ)-PTR for practical languages. In [48], Cramer et al. showed how to efficiently combine Σ-protocols in order to obtain a (k, ℓ)-PTR. Their construction mainly consists of running Σ-protocols for all instances combining them efficiently, and thus the costs (i.e., computations and communication) of their (k, ℓ)-PTR essentially consist of the sum of the costs of all the underlying Σ-protocols. The resulting protocol is still a Σ-protocol. More recently, a different technique has been proposed in [45] where Ciampi et al. showed how to obtain a similar result with the additional feature of postponing the need to know the instances to the last round (i.e., delayed input). The delayed-input (k, ℓ)-PTR of Ciampi et al. relies on the DDH assumption and can be applied to all Σ-protocols as [48]. The resulting protocol is a 3-round public-coin proof of knowledge. Unfortunately, since this composition technique relies on a computationally-hiding commitment scheme, it produces a protocol which only achieves computational zero knowledge regardless of the underlying Σ-protocols being statistical/perfect zero knowledge. However, statistical/perfect zero knowledge is very important since it protects the privacy of past proof computed by the prover forever (e.g., even if quantum computers become a concrete threat).

Very recently, Attema et al. in [7], improving a prior work of Groth and Kohlweiss [70], have shown how to obtain a very compact (k, ℓ)-PTR. However, the result of [7] only works for discrete logarithms (and variations), thus remaining far from the general results of [48]. Their construction requires a logarithmic number of rounds[1] and is secure against polynomial-time adversarial provers only (while preserving statistical/perfect zero knowledge). They require a shared random string (SRS) as trusted parameters.

Even more recently, Goel et al. in [64] have broken the barrier of linear (in ℓ) communication complexity when composing generic Σ-protocols, showing an efficient composition for a large class of Σ-protocols (which they call stackable Σ-protocols) obtaining logarithmic communication complexity. Their construction is secure against

[1]The result of [70] instead works only for $k = 1$, but it just requires 3 rounds.

polynomial-time adversarial provers only[2], obtaining computational special soundness. They give an instantiation of their construction based on a commitment scheme that relies on the discrete logarithm assumption. Their instantiation requires as trusted parameters the description of a collision-resistant hash function (CRHF) and parameters for Pedersen commitments. The perfect hiding of the commitment scheme allows to preserve statistical/perfect zero knowledge when the underlying Σ-protocols are perfect/statistical zero-knowledge. While their construction applies to a large class of Σ-protocols, the techniques of [64] are communication-efficient only when $k = 1$. In the full version of their paper [62], Goel et al. discuss (see Sec. 9.1 and App. F of [62]) an approach for the case of $k > 1$ but unfortunately, as they acknowledge, their proposal strongly affects communication, without providing substantial improvements over [48]. Goel et al. left explicitly open the problem of efficiently combining Σ-protocols in order to break, for generic values of k, the linear (in ℓ) barrier achieved by [48] (see [62], page 32, Sec. 9.1).

Alternative approaches. Recently, in [77] the result of [76] has been extended to (k, ℓ)-PTRs retaining the same communication advantage of [76] while optimizing computation efficiency. Their approach has communication complexity proportional to k times the longest branch. Mac'n'Cheese [18] is an interactive commit-and-prove zero-knowledge proof system for binary and arithmetic circuits. The communication complexity is proportional to k times the longest circuit plus an additive term which is logarithmic in ℓ. As commitments it uses information-theoretic MACs based on vector oblivious linear evaluation (VOLE). Note that Mac'n'Cheese is inherently private coin since the soundness relies on the verifier keeping the MAC key secret. Therefore, it is not immediately clear whether it can be modified to support public verifiability. Finally, one might leverage succinct proof techniques such as STARKs [22] or SNARKs [92] to get a communication-efficient (k, ℓ)-PTR. While these techniques can

[2]For ease of presentation, we will use the term PTR even when the soundness property holds only against a computationally bounded adversarial prover. We will do the same for computational Σ-protocols which only satisfy a weaker version of special soundness called computational special soundness (cfr., Sec. 2.3.1.1).

achieve even constant proof size, they have several drawbacks such as a huge workload for provers and the use of strong assumptions and/or problematic trusted setups. Although all the approaches above apply to NP-complete languages, we note that they are not so obviously efficiently generalizable to arbitrary languages. Instead, approaches based on composing Σ-protocols are more beneficial to protocol designers. Indeed, if there is a Σ-protocol (with specific additional properties) for the base relation, a protocol designer can use the composition technique directly for the (k, ℓ) case without the need to run a possibly expensive NP-reduction.

Open problem. In light of the above state of affairs, we have the following natural and interesting (both theoretically and practically) open question:

Is it possible to obtain practical (i.e., round, communication and computation-efficient) (k, ℓ)-proofs of knowledge for threshold relations for a large class of Σ-protocols (and thus for several useful languages) with communication complexity sublinear in ℓ preserving statistical/perfect zero knowledge?

4.1.2 Proofs over Threshold Relations: Our Contribution

In this work, we solve the above open problem when $k = o(\frac{\ell}{\log \ell})$ by showing how to efficiently combine the same large class of Σ-protocols considered in [62] obtaining a (k, ℓ)-PTR with communication complexity that is roughly[3] $k \log \ell$. In scenarios where k is way smaller than ℓ (e.g., k is constant or even $\sqrt{\ell}$) this is a significant improvement. Moreover, our construction, similarly to [62], can also be used for (k, ℓ)-PTR involving Σ-protocols for different languages. The protocol obtained through our techniques is still a Σ-protocol and thus, it can be combined again with our techniques or other techniques (e.g., [48]) for composing Σ-protocols. Finally, our construction preserves the flavour of the zero-knowledge property of the composed

[3]We will be more precise later making the impact of the security parameter explicit.

protocols. Indeed, our (k, ℓ)-PTR called $\Pi_{\mathsf{k},\ell}$ is still statistical/perfect honest-verifier zero knowledge (HVZK) if the base Σ-protocols are statistical/perfect HVZK. Our construction can also be instantiated, with small modifications, using the commitment scheme of [45]. In this case, our (k, ℓ)-PTR would provide computational HVZK but it would only require a shared CRHF as setup, that is a milder setup compared to requiring a shared random string.

We use the $(1, \ell)$-PTR of [62], to which we refer as $\Pi_{1,\ell}$, as a building block and start with their observation that repeating k times their construction is insecure since an adversarial prover might succeed in using a witness for the same instance in all the k executions. This is precisely the problem left unsolved in [62] for (k, ℓ)-PTRs that we solve in this paper.

Compact proof of consistency of commitment parameters. In $\Pi_{1,\ell}$, the use of a witness is associated to $\log \ell$ pairs of parameters of a commitment scheme for pairs of messages such that, for every pair, one parameter allows for equivocation and the other parameter prevents equivocation. We will say that one parameter is equivocal and the other one is binding. Only the prover knows which element is equivocal for every pair. We say that a commitment scheme is 1-out-of-2 equivocal when a commitment phase requires to commit to two messages, one with binding parameters and one with equivocal parameters, without allowing the receiver to distinguish them, even after the commitment is opened.

In the following, we assume w.l.o.g. that ℓ is a power of 2 and give a simplified description of our approach. Let $x_0, \ldots, x_{\ell-1}$ be the instances and let x_i with $i \in \{0, \ldots, \ell - 1\}$ be the instance corresponding to witness w_i known to the prover. The $\log \ell$ pairs of parameters are chosen so that the j-th pair has the first parameter binding if the j-th bit of i is zero (for $j = 0, \ldots, \log \ell - 1$) and equivocal otherwise. Notice that the connection between a pair of parameters that can be either (equivocal,binding) or (binding,equivocal) and a bit of the index of an instance makes the $\log \ell$ pairs of parameters associated to x_i logically different from the $\log \ell$ pairs of parameters associated to x_j, as long as $i \neq j$. We observe that to show that in k executions of $\Pi_{1,\ell}$ the k witnesses correspond to k different instances, one can focus on showing that the k sequences of $\log \ell$ pairs of parameters are all disjoint in the sense that for every pair $(\bar{a} = a_0, a_1, \ldots, a_{\log \ell - 1}), (\bar{b} = b_0, b_1, \ldots, b_{\log \ell - 1})$ of elements in these k sequences, there is always a position $j \in \{0, \ldots, \log \ell - 1\}$ such that only one out of a_j and b_j has the first parameter that is binding. We focus on efficiently proving the above property of all pairs in these k sequences as follows: first, we require the prover to sort the k sequences according to the order relation derived by assigning to every sequence of $\log \ell$ pairs of parameters a string of $\log \ell$ bits where the j-th bit is 0 if and only if the j-th pair of parameters in the sequence has the first element that is binding. The prover has all the information to sort these k sequences since the prover itself decided those parameters and thus it knows which one is binding and which one is equivocal. Once the k sequences are sorted, in order to prove that they are all disjoint (in the sense explained above) it suffices for the prover to show that for every two consecutive elements $(\bar{a}, \bar{b})$ in such ordered sequence of k elements, the bit representation of $\bar{b}$ is greater than the one of $\bar{a}$. With such trick, the prover must provide $k - 1$ proofs in total to show that all sequences are different. Each of such proofs is about proving a property of the involved $4 \log \ell$ parameters[4]. We show a concrete and efficient instantiation for the parameters of the 1-out-of-2 equivocal commitment scheme GGHK from [62]. The communication complexity of each of the above $k - 1$ proofs is $\mathcal{O}(\log \ell)$. In Table 4.1, we compare our results to the previously discussed approaches to obtain (k, ℓ)-PTRs. For the sake of completeness, in this comparison we make the security parameter λ explicit. Notice that our approach, while being less communication-efficient than [7, 70], is more flexible since it applies to a much wider family of languages.

Threshold ring signatures. As pointed out in [62], by making $\Pi_{1,\ell}$ non-interactive with the aid of a random oracle one gets a ring signature whose size is logarithmic in the size of the ring ℓ (see [62] Page 4 and Sec. 9.3). Following a similar approach, starting from our $\Pi_{\mathsf{k},\ell}$ we can get a threshold ring signature scheme according to Def. 3 of [75], but considering PPT adversaries instead of quantum polynomial-time

[4]For each proof there are two sequences of $\log \ell$ pairs of parameters.

adversaries. In a threshold ring signature scheme, k signers cooperate to sign a message hiding their identities within a larger group of size ℓ. In our threshold ring signature scheme, the size of a signature corresponds roughly to $\mathcal{O}(k \log \ell)$ group elements. Interestingly, while featuring a relatively simple design, our construction improves in terms of signature size on many literature works [30,93,122]. Other schemes have signature size which is linear in ℓ (while being independent of k) and thus are also outperformed when $k << \ell$ [43,75,128]. When comparing our construction to others achieving more compact signature sizes [8,75,90], our construction still has interesting advantages in terms of resilience to adversarially chosen keys (see [75]) or used assumptions. An additional feature of our threshold ring signature is that it requires a reduced level of interaction. The signers just have to interact with one party called the aggregator. After having interacted with all the signers, the aggregator just compiles all the received contributions into one threshold ring signatures which can be publicly posted. Finally, our techniques allow threshold ring signatures with more advanced hiding properties. Indeed, since our $\Pi_{\mathsf{k},\ell}$ is still a stackable Σ-protocol, it can be used as a base Σ-protocol for our threshold ring signature, thus expressing more complex relations with better anonymity properties (i.e., "threshold-of-threshold"). In particular, they can express hierarchical relations such as "k out of ℓ groups signed a message, and within each group $i \in [\ell]$ of size ℓ_i, at least k_i members signed the message". The number of levels of the hierarchy is arbitrary.

4.1.3 Technical Overview of [62]

We first describe 1-out-of-2 equivocal commitments (see Sec. 4.1.4.2 for more details) that are a major tool used in [62]. Then, we show how [62] exploits 1-out-of-2 equivocal commitments to get a $(1,\ell)$-PTR.

1-out-of-2 equivocal commitments in a nutshell. A 1-out-of-2 equivocal commitment allows a sender to commit to two values one of which is guaranteed to be binding, either unconditionally or under computational assumptions. The other element instead can be equivo-

Protocol	**Rounds**	**Communication**	**Values of k**	**Language**
[48]	3	$\mathcal{O}(\ell\mathsf{CC}(\Sigma))$	Any k	All Σ
[70]	3	$\mathcal{O}(\lambda \log \ell)$	$k = 1$	DL-like
[7]	$\mathcal{O}(\log \ell)$	$\mathcal{O}(\lambda \log(2\ell - k))$	Any k	DL-like
[62]	3	$\mathcal{O}(\mathsf{CC}(\Sigma) + \lambda \log \ell)$	$k = 1$	Stackable Σ
[62]	3	$\mathcal{O}(k(\mathsf{CC}(\Sigma) + \lambda\ell))$	$k > 1$	Stackable Σ
Ours	3	$\mathcal{O}(k(\mathsf{CC}(\Sigma) + \lambda \log \ell))$	Any k	Stackable Σ

Table 4.1: Comparison of several techniques for (k, ℓ)-PTR. When comparing more language-generic techniques like ours and [48,62], we express the communication complexity both in terms of the communication complexity of the underlying Σ-protocol $\mathsf{CC}(\Sigma)$ and of the security parameter λ. Notice that the communication complexity of [70] does not depend on k since their technique only works for $k = 1$. The Language column reports the languages or the class of the Σ-protocols for which the corresponding composition technique works. Despite being the least communication efficient, [48] can be applied to a wider class of Σ-protocols.

cated using a trapdoor that is known to the sender. Once the commitment is opened, the commitment scheme itself would guarantee that the equivocal position is not leaked. Before sending the commitment to the receiver, the commitment scheme parameters are generated by the sender who has to decide which position is equivocal. From now on, we call non-trapdoor (NT) a parameter that is associated with a binding position, while a trapdoor parameter (T) is associated with an equivocal position.

$(1, \ell)$-PTR through Σ-protocols. For simplicity, we will focus on instances belonging to the same language. Nevertheless, both in [62] and in our results it is possible to go beyond this restriction (see Sec. 8 of [62]).

The main idea in [62] is that every involved Σ-protocol has a deterministic HVZK simulator, called Extended HVZK (EHVZK) simulator which, given a challenge c, a third-round message z, and a statement x, outputs a simulated a such that (a, c, z) is an accepting transcript for the instance x. With EHVZK in their hands, the authors introduce the notion of stackable Σ-protocols. A Σ-protocol is stackable if (i) it has an EHVZK simulator and (ii) the third-round message is recyclable, meaning that the distribution of such messages is independent of the instance for every instance in the language.

Let us first consider just two of the ℓ instances, say x_1 and x_2. Given two executions Σ_1 and Σ_2 of a stackable Σ-protocol Π for instances x_1 and x_2 respectively, an execution $\Sigma_{1,2}$ of the composed Σ-protocol $\Pi_{1,2}$ defined by Goel et al. [62] for $x_1 \vee x_2$ can be constructed as follows. Let us assume that the prover $\mathsf{P}_{1,2}$ knows the witness corresponding to x_1. We name a_1 (respectively a_2) the first-round message of the underlying execution Σ_1 (respectively Σ_2), a the first-round message of the execution $\Sigma_{1,2}$ of $\Pi_{1,2}$, c the challenge sampled by the verifier $\mathsf{V}_{1,2}$ for $\Pi_{1,2}$, and z the last message of $\Sigma_{1,2}$. Since Σ_1 and Σ_2 are executions of the stackable Σ-protocol Π, their third-round messages have the same distribution. Therefore, the accepting third-round message from Σ_1 can be re-used as a third-round message for Σ_2 as described in the composed Σ-protocol $\Pi_{1,2}$ below:

- $\mathsf{P}_{1,2}$ computes the first-round message a_1 of protocol Π on input the instance x_1 and witness w_1. $\mathsf{P}_{1,2}$ commits to a_1 using a 1-out-of-2 equivocal commitment scheme. The value a_1 is put in the binding position, while the equivocal position commits to 0. We denote the resulting commitment as `com`. The first-round message a in the execution $\Sigma_{1,2}$ of the composed protocol $\Pi_{1,2}$ includes `com` as well as the parameters of the commitment scheme.

- Upon receiving the challenge c from $\mathsf{V}_{1,2}$, $\mathsf{P}_{1,2}$ computes z' using witness w_1, and equivocates the equivocal position of the commitment with a simulated a_2. The value a_2 is obtained by running the EHVZK simulator of Π with input the instance x_2, c, and the value z' computed above. Then, $\mathsf{P}_{1,2}$ sends z' and the opening values of `com` to $\mathsf{V}_{1,2}$ as a third-round message z of $\Sigma_{1,2}$. The value z also includes the commitment parameters.

- $\mathsf{V}_{1,2}$ reconstructs a_1 and a_2 by running the EHVZK simulator of Π. Then, $\mathsf{V}_{1,2}$ checks that both (a_1, c, z') and (a_2, c, z') are accepting transcripts for V_1 and V_2, and that `com` actually opens to a_1 and a_2.

Since $\Pi_{1,2}$ is still a stackable Σ-protocol, it can be recursively used to prove the instance $x_1 \vee x_2 \vee x_3 \vee x_4$. Indeed, this can be seen again as an OR of two statements, therefore the Σ-protocol for the instance $(x_1 \vee x_2) \vee (x_3 \vee x_4)$ can be composed using the same technique. Then, one can iterate the same process to obtain a $(1, 8)$-PTR by applying the same technique to two $(1, 4)$-PTR, and so on. Such composition of ℓ disjunctive instances can be represented by the following binary tree: the leaves of the tree represent the ℓ base executions $(\Sigma_1, \dots, \Sigma_\ell)$ of the Σ-protocol Π. Given two siblings nodes i and j, with associated protocol execution Σ_i for the instance x_i and Σ_j for the instance x_j respectively, the parent node t of i and j describes the execution of the protocol Σ_t obtained by applying the compiler for $(1, 2)$-PTR of [62]. Moreover, edges (t, i) and (t, j) are labeled as follows: if P_t knows a witness for the instance x_i, then the edge (t, i) is labelled with NT to indicate that, in the commitment computed by P_t in the first round, the position where x_i is used is binding. The edge (t, j) is labeled with T to indicate that the position where x_j is used is equivocal. If P_t knows a witness for x_j instead, then the opposite holds. An example of a tree induced by recursively applying the composition of [62] for a $(1, 2)$-PTR to get a $(1, 8)$-PTR is shown in Fig. 4.1. This recursive application of the (1,2)-PTR gives a communication complexity for the $(1, \ell)$-PTR that is roughly logarithmic in ℓ.

(k, ℓ)-PTR extension. In [62], an extension of their compiler to achieve a (k, ℓ)-PTR[5] is proposed. They propose a k-out-of-ℓ binding vector-of-vectors commitment scheme. This modification allows to equivocate at most $\ell - k$ positions. Roughly speaking, they instantiate such a primitive by making the commit algorithm output a matrix of $k \times \ell$ commitment values (i.e., each row is a 1-out-of-ℓ equivocal commitment), together with a non-interactive zero-knowledge proof that the binding position is different in each row. As pointed out in [62], with this technique they lose their ability to recursively apply the $(1, 2)$-PTR compiler. As a result, their (k, ℓ)-PTR has a communication complexity of roughly $\mathcal{O}(k\ell)$ (see Table 4.1).

[5]See Sec. 9.1 and App. F of [62].

4.1.4 Our Techniques

Our approach for a communication-efficient (k, ℓ)-PTR. In [62], the location of the instance for which the prover knows the witness uniquely determines the way parameters are laid out over the composition tree. For example, in Fig. 4.1 the instance for which $\mathsf{P}_{1,2}$ knows the witness is x_1. This means that, starting from $\Sigma_{1,2}$, the position containing the first message of Σ_1 has to be binding. Indeed, since $\mathsf{P}_{1,2}$ only holds a witness for x_1, $\mathsf{P}_{1,2}$ is able to produce an accepting transcript exclusively for Σ_1. Therefore, the third-round message to be recycled has to come from Σ_1, while the committed first-round message of the execution Σ_2 needs to be equivocated with the output of the EHVZK simulator. It follows that, climbing up the tree, the commitment position containing the first message of $\Sigma_{1,2}$ has to be binding. Indeed, $\mathsf{V}_{1,2,3,4}$ will in turn execute $\mathsf{V}_{1,2}$ and $\mathsf{V}_{3,4}$, which internally use the verifiers of the base Σ-protocols. This means that in $\Sigma_{3,4}$ the prover recycles the third-round message of $\Sigma_{1,2}$ and that in $\Sigma_{1,2,3,4}$ the committed first-round message of $\Sigma_{3,4}$ has to be equivocated accordingly in order to get an accepting transcript. Applying the same reasoning again, it is easy to conclude that in $\Sigma_{1,\ldots,8}$ the binding position of the 1-out-of-2 equivocal commitment is again the same.

A crucial idea of [62] to achieve logarithmic communication complexity is reusing commitment parameters and openings across the same levels of the composition tree. The composition is designed so that commitment parameters and openings are part of the third-round message of the composed protocol. Indeed, since the composed Σ-protocol of [62] is itself stackable, it follows that its EHVZK simulator takes as input commitment parameters and openings to generate a suitable first-round message, namely a 1-out-of-2 equivocal commitment reusing the same openings and parameters[6]. This means that since all the Σ-protocols executions that belong to the same level of

[6]For this composition to work and to compress the communication complexity down to logarithmic, the size of the equivocal commitment must be independent of the size of the committed value. To solve this issue, committed values have to be compressed down to a constant size with the aid of a collision-resistant hash function.

the tree share the same third-round message, they also have to use the exact same commitment parameters. Therefore, in the $(1,\ell)$-PTR of [62] given the instance x_i corresponding to the witness used by the prover, there is a unique way in which commitment parameters can be laid out over the composition tree. Thus, to build a (k,ℓ)-PTR it suffices to repeat the construction of [62] k times and to prove that the composition trees of such k executions are all distinct. In this section we describe how we design a communication-efficient Σ-protocol for the above goal.

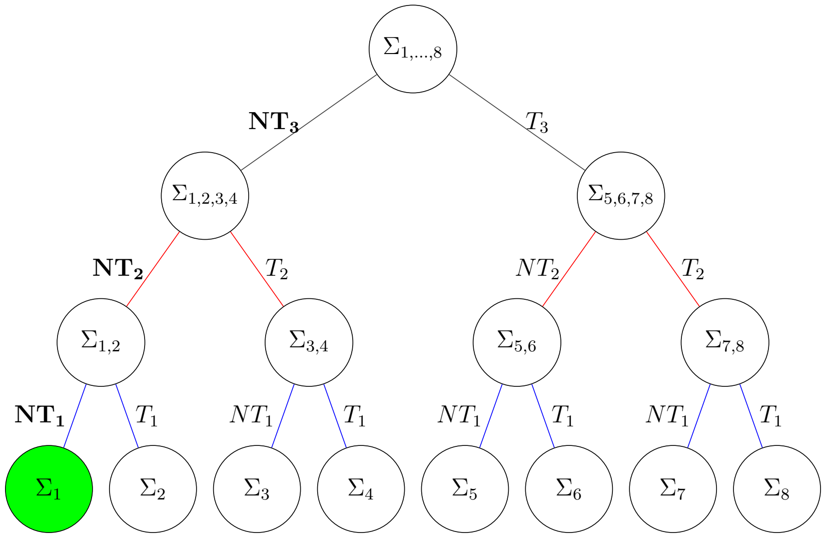

Figure 4.1: An example of a composition tree induced by the recursive application of the $(1,2)$-PTR of [62] in which 8 base Σ-protocols are composed to obtain a $(1,8)$-PTR. In this example, $\mathsf{P}_{1,\dots,8}$ knows a witness for the instance x_1. This implies that, going from the root to the leaves, the left-most branch must be non-trapdoor. Additionally, commitment openings and parameters are re-used across the same level of the composition tree and this is emphasized by using the same index and the same color for all the edges within a level.

4.1.4.1 Stackable Σ-protocols

Let us consider the notion of Σ-protocol as a 3-round public coin protocol $\Pi = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$ with the properties defined in Sec. 2.3.1.1. Traditionally, Σ-protocols enjoy the special soundness property with a perfect flavour. However, for convenience, in the remainder of the paper we will use the term Σ-protocols also to refer to protocols that just enjoy computational special soundness.

We now define the stackable Σ-protocol of [62].

Definition 19 (Computational/Statistical EHVZK)**.** *Let* $\Sigma = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$*, be a* Σ*-protocol for an NP language* $\mathcal{L}$*.* Σ *is EHVZK if there exists a PPT algorithm* $\mathcal{S}^{\mathsf{EHVZK}}$ *such that for all PPT/unbounded* D*, and* $c \in \{0,1\}^\lambda$*, there exists an efficiently samplable distribution* $D^{(z)}_{x,c}$ *and a negligible function* $\nu(\cdot)$ *such that for all* $x \in \mathcal{L}$

$$\left| \Pr\left[\mathsf{ExpEHVZK}_{(\mathsf{P}_0,\mathsf{P}_1),\mathsf{D}}(c) = 1\right] - \Pr\left[\mathsf{ExpEHVZK}_{\mathcal{S}^{\mathsf{EHVZK}},\mathsf{D}}(c) = 1\right] \right| \leq \nu(|x|).$$

We say instead that a Σ-protocol is perfect EHVZK if the difference between the probabilities is exactly 0. The experiment $\mathsf{ExpEHVZK}$ for EHVZK follows.

$\mathsf{ExpEHVZK}_{\mathsf{P}',\mathsf{D}}(c)$

1. $(x, w) \leftarrow \mathsf{D}(c)$.
2. If $(x, w) \notin \mathcal{R}_{\mathcal{L}}$, return 0.
3. If $\mathsf{P}' = \mathcal{S}^{\mathsf{EHVZK}}$, sample $z \leftarrow_\$ D^{(z)}_{x,c}$ and compute $a \leftarrow \mathcal{S}^{\mathsf{EHVZK}}(x, c, z)$.
4. Otherwise, sample $R \leftarrow_\$ \{0,1\}^\lambda$, compute $a \leftarrow \mathsf{P}_0(x, w; R)$ and $z \leftarrow \mathsf{P}_1(x, w, a, c; R)$.
5. Return $\mathsf{D}(x, w, a, c, z)$.

Definition 20 (Σ-protocol with recyclable third messages)**.** *Let* $\Sigma = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$ *be a* Σ*-protocol for an NP language* $\mathcal{L}$*,* Σ *has recyclable third messages if for every* $c \in \{0,1\}^\lambda$*, there exists an efficiently samplable distribution* $D^{(z)}_c$*, such that for all* $(x, w) \in \mathcal{R}_{\mathcal{L}}$*, it holds that* $D^{(z)}_c \approx \{z | R \leftarrow_\$ \{0,1\}^\lambda; a \leftarrow \mathsf{P}_0(x, w; R); z \leftarrow \mathsf{P}_1(x, w, c; R)\}$.

Definition 21 (Stackable Σ-protocol). *We say that a Σ-protocol $\Sigma = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$, is stackable, if it is a EHVZK Σ-protocol and has recyclable third messages.*

4.1.4.2 1-out-of-2 Equivocal Commitment Schemes

We now define the notion of 1-out-of-2 equivocal commitment scheme. The sender commits to a pair of messages (m_0, m_1) with $m_0, m_1 \in \{0,1\}^\lambda$, where $\lambda \in \mathbb{N}$ is the security parameter. A 1-out-of-2 equivocal commitment scheme $CS = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{BindCom}, \mathsf{EquivCom}, \mathsf{Equiv}, \mathcal{R}_T)$ consists of five PPT algorithms and a polynomial-time relation $\mathcal{R}_T$. The algorithm Setup generates a common reference string pp. We denote by $\mathcal{Y}^{\mathsf{pp}}$ the space of well-formed commitment parameters w.r.t. pp and require that membership in $\mathcal{Y}^{\mathsf{pp}}$ can be checked efficiently. The above algorithms work as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda; r)$: on input the security parameter, and randomness r, generates public parameters pp.
- $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta; r)$: on input public parameters pp, binding position $\beta \in \{0,1\}$, and randomness r, returns the commitment parameters $(p_0, p_1) \in \mathcal{Y}^{\mathsf{pp}}$ and the trapdoor td for parameter $p_{1-\beta}$ such that $(p_{1-\beta}, \mathsf{td})$ belongs to $\mathcal{R}_T$[7].
- $\mathtt{com} \leftarrow \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0, m_1; r)$: on input public parameters pp, commitment parameters p_0, p_1, messages m_0, m_1, and randomness r outputs a commitment $\mathtt{com}$.
- $(\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta, m, p_0, p_1, \mathsf{td}; r)$: on input public parameters pp, binding position β, message of the binding position m, commitment parameters p_0, p_1, trapdoor td, and randomness r returns a commitment $\mathtt{com}$ and auxiliary information aux.
- $r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, m_0, m_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$: on input public parameters pp, binding position β, messages m_0, m_1, commitment pa-

[7]The statement for $\mathcal{R}_T$ may also depend from pp. We will omit this dependence to simplify the notation.

rameters p_0, p_1, trapdoor td, and auxiliary information aux, deterministically returns an equivocation randomness r.

In the following, we assume that pp was already generated by a trusted third party using the algorithm Setup. Furthermore, we will omit the randomness from the input of the algorithms, except when it is relevant. A sender and a receiver interact using the commitment scheme as follows.

Commit Phase: The sender, on input m and binding position β, computes $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta)$, $(\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta, m, p_0, p_1, \mathsf{td})$. The sender sends $(\mathtt{com}, p_0, p_1)$ to the receiver.

Reveal Phase: The sender, on input m^*, computes $r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, m_0, m_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$ where $m_\beta = m$ and $m_{1-\beta} = m^*$, and sends (r, m_0, m_1) to the receiver. The receiver computes $\mathtt{com}' \leftarrow \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0, m_1; r)$ and accepts if $\mathtt{com}' = \mathtt{com}$ and $(p_0, p_1) \in \mathcal{Y}^{\mathsf{pp}}$, rejects otherwise.

We state below the properties we require for the 1-out-of-2 equivocal commitment scheme.

Partial Equivocation: For all $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\beta \in \{0,1\}$, $(p_0, p_1) \in \mathcal{Y}^{\mathsf{pp}}$, $(m_0, m_1) \in \{0,1\}^{2\lambda}$, td such that $(p_{1-\beta}, \mathsf{td}) \in \mathcal{R}_T$ the following holds:

$$\Pr\left[\begin{array}{c|c} \begin{array}{c}\mathsf{BindCom}(\mathsf{pp}, p_0, p_1, \\ m_0, m_1; r) = \mathtt{com}\end{array} & \begin{array}{c}(\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\\ \mathsf{pp}, \beta, m_\beta, p_0, p_1, \mathsf{td}); \\ r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, m_0, \\ m_1, p_0, p_1, \mathsf{td}, \mathsf{aux})\end{array}\end{array}\right] = 1.$$

Computational Fixed Equivocation: Given the experiment $\mathsf{ExpFixEquiv}$ below, for every non-uniform PPT $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for any $\lambda \in \mathbb{N}$: $\Pr[\mathsf{ExpFixEquiv}_{\mathcal{A}}(\lambda) = 1] \leq \nu(\lambda)$.

```
$\mathsf{ExpFixEquiv}_{\mathcal{A}}(\lambda)$

    1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
    2. $(p_0, p_1, r^1, r^2, r^3, r^4,$
       $m_0^1, m_0^2, m_1^1, m_1^2, m_0^3, m_0^4, m_1^3, m_1^4) \leftarrow \mathcal{A}(\mathsf{pp})$.
    3. Return 1 if $\exists \beta \in \{0,1\}$ such that

       $(\mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0^1, m_1^1; r^1) = \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0^2, m_1^2; r^2) \wedge$
       $(\mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0^3, m_1^3; r^3) = \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0^4, m_1^4; r^4) \wedge$
       $(m_{1-\beta}^1 \neq m_{1-\beta}^2) \wedge (m_\beta^3 \neq m_\beta^4) \wedge ((p_0, p_1) \in \mathcal{Y}^{\mathsf{pp}})$.

       Return 0 otherwise.
```

Moreover, the protocol achieves perfect fixed equivocation if for any unbounded $\mathcal{A}$ it holds that $\Pr[\mathsf{ExpFixEquiv}_{\mathcal{A}}(\lambda) = 1] = 0$.

Computational Position Hiding: Given the experiment ExpHid below, for every non-uniform PPT $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for any $\lambda \in \mathbb{N}$:
$\Pr[\mathsf{ExpHid}_{\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \nu(\lambda)$.

```
$\mathsf{ExpHid}_{\mathcal{A}}(\lambda)$

    1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
    2. Sample $\beta \leftarrow\$ \{0,1\}$ and compute $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta)$.
    3. $\beta' \leftarrow \mathcal{A}(\mathsf{pp}, p_0, p_1)$.
    4. Return 1 if $\beta' = \beta$ and 0 otherwise.
```

Moreover, if $\mathcal{A}$ is unbounded and $\nu(\lambda) = 0$ we say that the scheme is perfect position hiding.

Computational Trapdoorness: Given the experiment $\mathsf{ExpTrap}$, for every non-uniform PPT $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for any $\lambda \in \mathbb{N}$:
$\Pr[\mathsf{ExpTrap}_{\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \nu(\lambda)$.

$\mathsf{ExpTrap}_{\mathcal{A}}(\lambda)$

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(m_0, m_1, p_0, p_1, \mathsf{td}, \beta) \leftarrow \mathcal{A}(\mathsf{pp})$.
3. If $(p_0, p_1) \notin \mathcal{Y}^{\mathsf{pp}}$ or $(p_{1-\beta}, \mathsf{td}) \notin \mathcal{R}_T$ abort the experiment.
4. Sample $b \leftarrow\!\!\$\, \{0,1\}$. If $b = 0$, set $(\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta, m_\beta, p_0, p_1, \mathsf{td})$ and set $r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, m_0, m_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$. If $b = 1$, sample $r \leftarrow\!\!\$\, \mathtt{D}$ and set $\mathtt{com} \leftarrow \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0, m_1; r)$.
5. $b' \leftarrow \mathcal{A}(\mathsf{pp}, m_0, m_1, p_0, p_1, \mathsf{td}, \beta, \mathtt{com}, r)$.
6. Return 1 if $b = b'$, return 0 otherwise.

Moreover, if $\mathcal{A}$ is unbounded and $\nu(\lambda) = 0$ we say that the protocol achieves perfect trapdoorness.

GGHK enjoys computational fixed equivocation, perfect position hiding and perfect trapdoorness[8]. Moreover, since GGHK has perfect position hiding and perfect trapdoorness, the use of this commitment scheme in $\Pi_{1,\ell}$ preserves the statistical/perfect zero knowledge property of the underlying Σ-protocols. Computational fixed equivocation forces the special soundness to be only computational instead (special soundness is degraded in any case since in [62] a hash function is used to compress the size of first-round messages).

[8]Our definition slightly differs from the one in [62]. In particular, our fixed equivocation property implies the partial binding of [62]. We need this slightly stronger property to prove the soundness of our (k, ℓ)-PTR. Natural instantiations such as GGHK enjoy the fixed equivocation property. The remaining properties are just a restatement of the minimal requirements for a 1-out-of-2 commitment scheme in [62].

4.1.4.3 Π_{ord}: a Σ-Protocol to Prove Parameters Ordering

We use the following notation: for any vector $\mathbf{v}$, $\mathbf{v}[z]$ indicates the z-th element of the vector $\mathbf{v}$. The first element of a vector $\mathbf{v}$ is indexed as $\mathbf{v}[1]$. Moreover, we use $[n]$ for $n \in \mathbb{N}$ to identify the set $\{1, \dots, n\}$. Let $\mathbf{x} = ((p_0^1, p_1^1), \dots, (p_0^n, p_1^n))$ be a vector containing n pairs of parameters corresponding to n instantiations of a 1-out-of-2 equivocal commitment scheme, where p_0^i represents the parameter of the first position of the i-th commitment instantiation, and p_1^i is the analogue for the second position. Consider the relations $\mathcal{R}_{T_0}$ and $\mathcal{R}_{T_1}$, where $\mathcal{R}_{T_0} = \{(x = (p_0, p_1), w) : p_1$ is a trapdoor parameter and w is the corresponding trapdoor$\}$ and, similarly $\mathcal{R}_{T_1} = \{(x = (p_0, p_1), w) : p_0$ is a trapdoor parameter and w is the corresponding trapdoor$\}$ (i.e., $(x = (p_0, p_1), w) \in \mathcal{R}_{T_i}$ if and only if $(p_{1-i}, w) \in \mathcal{R}_T$ with $i \in \{0, 1\}$). We present a Σ-protocol Π_{ord} that, given a vector $X = (\mathbf{x_1}, \dots, \mathbf{x_k})$ of vectors each containing n pairs of parameters corresponding to n instantiations of a 1-out-of-2 equivocal commitment scheme, allows a prover P to efficiently prove knowledge of a witness for $X \in \mathcal{L}$ where

$$\begin{aligned}\mathcal{L} = \{&X : \exists W = (\mathbf{w_1}, \dots, \mathbf{w_k}) \text{ such that} \\ &\forall\, i, j \in [k] \text{ with } i \neq j\ \exists z \in [n], \text{such that } b_{i,z} \neq b_{j,z}\}\end{aligned} \tag{4.1}$$

where $b_{i,z}, b_{j,z} \in \{0, 1\}$ satisfy $(\mathbf{x_i}[\mathbf{z}], \mathbf{w_i}[\mathbf{z}]) \in \mathcal{R}_{T_{b_{i,z}}}$, $(\mathbf{x_j}[\mathbf{z}], \mathbf{w_j}[\mathbf{z}]) \in \mathcal{R}_{T_{b_{j,z}}}$.

An efficient Σ-protocol. One could naively prove the above statement by separately proving that each vector of n pairs of commitment parameters differs in the way equivocal and binding parameters are laid out w.r.t. every other vector. Carrying out such proof would involve a quadratic (in k) amount of separate proofs. We instead take a different path, that is introducing a strict total ordering among these k vectors of n pairs of commitment parameters. In particular, we map a vector $\mathbf{x}$ of n pairs of commitment parameters to a binary string $s \in \{0, 1\}^n$ by setting $s = b_1 || \dots || b_n$, for which $(\mathbf{x}[\mathbf{z}], \mathbf{w}[\mathbf{z}]) \in \mathcal{R}_{T_{b_z}}$. Let s_m with $m \in [k]$ be the string resulting by applying the above mapping to a vector $\mathbf{x_m}$ of n pairs of commitment parameters. W.l.o.g. consider the case where $s_1 > \dots > s_k$. If the above order relation holds, it follows that all the k vectors of n pairs of commitment parameters are logically different from each other in terms of how the trapdoor parameters are laid out in at least one position. Notice that after having introduced such ordering among the k vectors of n pairs of commitment parameters, one can come up with the language $\mathcal{L}_{ord}$ described by only a linear number of comparisons. Indeed, the language of Equation 4.1 can be equivalently rewritten as $\mathcal{L}_{ord} = \{X : \exists W = (\mathbf{w_1}, \ldots, \mathbf{w_k})$ such that $s_1 > s_2 > \ldots > s_k\}$, where for all $m \in [k]$, $s_m = b_1 || \ldots || b_n$, for which $(\mathbf{x_m}[\mathbf{i}], \mathbf{w_m}[\mathbf{i}]) \in \mathcal{R}_{T_{b_i}}$.

Instantiation. Let us consider two binary strings $s_1 \in \{0,1\}^n$ and $s_2 \in \{0,1\}^n$, in which we use $s[i]$ to indicate the i-th bit of the string s, it is pretty straightforward to see that if $s_1 > s_2$, the following formula also holds[9]:

$$\bigvee_{i=1}^{n} \left(\left(\bigwedge_{j=0}^{i-1} (s_1[j] = s_2[j]) \right) \wedge (s_1[i] > s_2[i]) \right). \tag{4.2}$$

Indeed, this corresponds to performing a bit-wise comparison between s_1 and s_2, starting from the most significant bit. Namely, if $s_1 > s_2$, the first different bit between the two strings has value 1 in s_1 and 0 in s_2.

Building on this observation, we can construct a protocol $\Pi_{ord'}$ to prove that two binary strings, each representing a vector of n 1-out-of-2 equivocal commitment parameters, are such that one is greater than the other. Then, given k vectors of commitment parameters, one can prove that $X = (\mathbf{x_1}, \ldots, \mathbf{x_k}) \in \mathcal{L}_{ord}$, where $|\mathbf{x_i}| = n$ for all $i \in [k]$, by using $\Pi_{ord'}$ $k-1$ times. We denote the resulting protocol as Π_{ord}.

1-out-of-2-commitment of [62]. In [62], a t-out-of-ℓ equivocal commitment scheme GGHK based on the discrete logarithm assumption is defined. GGHK uses the same SRS of the non-interactive version of the Pedersen commitment scheme, namely, two generators g_0, h of a group $\mathbb{G}$ where the discrete logarithm of g_0 in base

[9]For consistency reasons, we assign the index 1 to the first position within the string and we say that $s_i[0] = 0$.

h is not known. For the interesting case of $t = 1$ and $\ell = 2$, the commitment parameters are two group generators algebraically derived from the SRS. The receiver is able to verify that the parameters are correctly generated via a simple algebraic check. In particular, $\mathcal{Y}^{\mathsf{pp}} = \{p_0 \in \mathbb{G}, p_1 \in \mathbb{G} : p_1 = p_0^2 g_0^{-1}\}$. The trapdoor associated to the equivocal position is the discrete logarithm in base h of the corresponding parameter. Thus, $\mathcal{R}_T = \{(x, w) : x = h^w\}$. We refer the reader to [62] for more details on the actual construction.

Instantiating Π_{ord} for the commitment of [62]. We now instantiate Π_{ord} for vectors of commitment parameters of GGHK. To do so, we just need to express Formula 4.2 in terms of the parameters of GGHK. Given (p_0^a, p_1^a), we represent membership of (p_b^a, w) in $\mathcal{R}_T$ as the function $\mathcal{R}_{\mathsf{DL}}(p_b^a, w)$ evaluating to 1 if w is the discrete logarithm of p_b^a w.r.t. h and 0 otherwise. Given two vectors of n commitment parameters of GGHK $P = ((p_0^1, p_1^1), \ldots, (p_0^n, p_1^n))$ and $Q = ((q_0^1, q_1^1), \ldots, (q_0^n, q_1^n))$, and two vectors of corresponding witnesses $W_p = (w_p^1, \ldots, w_p^n)$ and $W_q = (w_q^1, \ldots, w_q^n)$, Formula 4.2 can be rewritten as a relation $\mathcal{R}_{ord'}$ on input $((P, Q), (W_p, W_q))$[10]:

$$\bigvee_{i=1}^{n} \left(\left(\bigwedge_{j=0}^{i-1} (((\mathcal{R}_{\mathsf{DL}}(p_0^j, w_p^j) \wedge \mathcal{R}_{\mathsf{DL}}(q_0^j, w_q^j)) \vee (\mathcal{R}_{\mathsf{DL}}(p_1^j, w_p^j) \wedge \mathcal{R}_{\mathsf{DL}}(q_1^j, w_q^j))) \right) \right. \\ \left. \wedge \left(\mathcal{R}_{\mathsf{DL}}(p_1^i, w_p^i) \wedge \mathcal{R}_{\mathsf{DL}}(q_0^i, w_q^i)\right) \right). \quad (4.3)$$

Basically, for each bit of the strings s_1 and s_2 of Formula 4.2, such bits are equal if the corresponding parameters pairs have the same trapdoor position, meaning that either the sender knows the discrete log of both the first positions of the pairs, or that the same applies for the second position of both parameters pairs. A bit of a string is defined to be 1 if the corresponding parameters pair has in its first

[10]In the following formula the AND ranging from $j = 0$ to $j = i - 1$ is evaluated as true for $j = 0$. Indeed, according to the notation used in this paper, there are no parameters pair in the position 0 of the vector.

position a group element with a discrete log that is known to the sender, while it is 0 if the same applies to the second position.

Given k vectors $V_1, \dots, V_k$ of n pairs of commitment parameters and k vectors $W_1, \dots, W_k$ of witnesses, the relation proved by Π_{ord}, is defined as follows:

$$\mathcal{R}_{ord}((V_1, \dots, V_k), (W_1, \dots, W_k)) = \bigwedge_{i=1}^{k-1} \mathcal{R}_{ord'}((V_i, V_{i+1}), (W_i, W_{i+1})). \tag{4.4}$$

Our instantiation of Π_{ord} can be obtained via OR/AND compositions of the Schnorr's Σ-protocol [48], which is also stackable [62]. As a result, Π_{ord} is a stackable Σ-protocol with computational special soundness and perfect EHVZK. $\Pi_{ord'}$, proving $\mathcal{R}_{ord'}$, achieves communication complexity $\mathcal{O}(n\lambda + \lambda \log n) = \mathcal{O}(n\lambda)$. Π_{ord} can be obtained by repeating $\Pi_{ord'}$ $k-1$ times in parallel, obtaining a communication complexity of $\mathcal{O}((k-1)n\lambda) = \mathcal{O}(kn\lambda)$.

4.1.4.4 Efficient (k, ℓ)-PTR

We build our (k, ℓ)-PTR repeating the $(1, \ell)$-PTR of [62] k times and using Π_{ord} with statements the k vectors of commitment parameters of length $\mathcal{O}(\log \ell)$ that constitute the composition trees. Notice that Π_{ord} is defined over ordered tuples of pairs of commitment parameters. However, the prover can easily sort the k underlying $(1, \ell)$-PTRs according to such order. Let $(\mathsf{P}_0^{1,\ell}, \mathsf{P}_1^{1,\ell})$ be the prover algorithms of the $(1, \ell)$-PTR from [62]. W.l.o.g., we also assume that the algorithm $\mathsf{P}_0^{1,\ell}$ outputs, together with the first-round message a to be sent to $\mathsf{V}^{1,\ell}$, the tuple of commitment parameters $((p_0^1, p_1^1), \dots, (p_0^{\log \ell}, p_1^{\log \ell}))$ and the related witnesses tuple $(\mathsf{td}^1, \dots, \mathsf{td}^{\log \ell})$. In our (k, ℓ)-PTR $\Pi_{\mathsf{k},\ell} = (\mathsf{P}_0^{\mathsf{k},\ell}, \mathsf{P}_1^{\mathsf{k},\ell}, \mathsf{V}^{\mathsf{k},\ell})$, the prover takes as input a tuple of statements $\mathbf{x} = (x_1, \dots, x_\ell)$ and k witnesses $\mathbf{w} = ((w_1, \alpha_1), \dots, (w_k, \alpha_k))$ in which $\alpha_j \in [\ell]$ is the position of the j-th witness. $\Pi_{\mathsf{k},\ell}$ uses the $(1, \ell)$-PTR $\Pi_{1,\ell} = (\mathsf{P}_0^{1,\ell}, \mathsf{P}_1^{1,\ell}, \mathsf{V}^{1,\ell})$, and $\Pi_{ord} = (\mathsf{P}_0^{ord}, \mathsf{P}_1^{ord}, \mathsf{V}^{ord})$.

First Round: The prover invokes $\mathsf{P}_0^{\mathsf{k},\ell}$ that, on input $(\mathbf{x}, \mathbf{w}; \mathtt{rand})$ computes a as follows:

1. Parse $\mathtt{rand}$ as $\mathtt{rand}_{\mathsf{P}_1}||\ldots||\mathtt{rand}_{\mathsf{P}_k}||\mathtt{rand}_{ord}$;
2. For all $j \in [k]$: Run $(a_j, \mathbf{p}_j, \mathbf{td}_j) \leftarrow \mathsf{P}_0^{1,\ell}(\mathbf{x}, (w_j, \alpha_j); \mathtt{rand}_{\mathsf{P}_j})$, where $\mathbf{p}_j = ((p_0^{(1,j)}, p_1^{(1,j)}), \ldots, (p_0^{(\log \ell, j)}, p_1^{(\log \ell, j)}))$ and $\mathbf{td}_j = (\mathsf{td}^{(1,j)}, \ldots, \mathsf{td}^{(\log \ell, j)})$;
3. Generate $a_{ord} \leftarrow \mathsf{P}_0^{ord}((\mathbf{p}_j)_{j\in[k]}, (\mathbf{td}_j)_{j\in[k]}; \mathtt{rand}_{ord})$;

The prover sends $a = (a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$ to the verifier.

Second Round: The verifier samples $c \in \{0,1\}^\lambda$ and sends c to the prover.

Third Round: The prover invokes $\mathsf{P}_1^{\mathsf{k},\ell}$ that computes z as follows: For each $j \in [k]$ run $z_j \leftarrow \mathsf{P}_1^{1,\ell}(\mathbf{x}, (w_j, \alpha_j), c; \mathtt{rand}_{\mathsf{P}_j})$ and $z_{ord} \leftarrow \mathsf{P}_1^{ord}((\mathbf{p}_j)_{j\in[k]}, (\mathbf{td}_j)_{j\in[k]}, c; \mathtt{rand}_{ord})$;

Then, the prover sends $z = (z_1, \ldots, z_k, z_{ord})$ to the verifier.

Verification: The verifier invokes $\mathsf{V}^{\mathsf{k},\ell}$ that, on input $(\mathbf{x}, a = (a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k), c, z = (z_1, \ldots, z_k, z_{ord}))$, returns a bit b as follows:

1. For $i \in [k]$ and $j \in [\log \ell]$ check that $(p_0^{(i,j)}, p_1^{(i,j)}) \in \mathcal{Y}^{\mathsf{pp}}$, where the pairs $(p_0^{(i,j)}, p_1^{(i,j)})$ are taken from z_i;
2. Check that $\mathbf{p}_i = (p_0^{(i,j)}, p_1^{(i,j)})_{j\in[\log \ell]}$;
3. For all $i \in [k]$ check that $\mathsf{V}^{1,\ell}(\mathbf{x}, a_i, c, z_i) = 1$;
4. Check that $\mathsf{V}^{ord}((\mathbf{p}_j)_{j\in[k]}, a_{ord}, c, z_{ord}) = 1$;
5. If all the previous checks are successful, output 1. Otherwise, output 0.

Figure 4.2: Our communication-efficient (k, ℓ)-PTR from stackable Σ-protocols.

Considering the relation $\mathcal{R}_{\mathsf{k},\ell} = \{((x_1, \ldots, x_\ell), ((w_1, \alpha_1), \ldots, (w_k, \alpha_k)))|1 \leq \alpha_1 < \ldots < \alpha_k \leq \ell \land \forall\, j \in [k] : (x_{\alpha_j}, w_j) \in \mathcal{R}_\mathcal{L}\}$, we state the theorem below.

Theorem 1. *Let $\Pi_{1,\ell}$ be the stackable Σ-protocol of [62], and let Π_{ord} be the stackable Σ-protocol of Sec. 4.1.4.3. $\Pi_{\mathsf{k},\ell} = (\mathsf{P}^{\mathsf{k},\ell}, \mathsf{V}^{\mathsf{k},\ell})$ described in Fig. 4.2 is a stackable Σ-protocol for relation $\mathcal{R}_{\mathsf{k},\ell}$ with computational special soundness. Furthermore, $\Pi_{\mathsf{k},\ell}$ preserves the EHVZK flavour of the underlying $\Pi_{1,\ell}$.*

We remark that, since the EHVZK flavour of our Π_{ord} instantiation is perfect, Π_{ord} does not affect the EHVZK flavour of $\Pi_{\mathsf{k},\ell}$. We will go through the proof by proving lemmas for completeness, computational special soundness, and EHVZK.

Lemma 1 (Completeness)**.** *$\Pi_{\mathsf{k},\ell}$ is complete.*

Proof. It follows from the completeness of the underlying protocols. □

For the sake of clarity, we first report a sketch of the proof of the computational special soundness property. Then, we report a detailed proof of the theorem.

Computational special soundness proof sketch. To prove the computational special soundness of $\Pi_{\mathsf{k},\ell}$, we exploit the computational special soundness of the k executions of $\Pi_{1,\ell}$, the computational special soundness of Π_{ord}, as well as the partial equivocation and the fixed equivocation properties of the 1-out-of-2 equivocal commitment scheme. Let us first review the extractor $\mathsf{Extract}_{1,\ell}$ of $\Pi_{1,\ell}$. $\mathsf{Extract}_{1,\ell}$ works via recursive calls to $\mathsf{Extract}_{1,2}$, the extractor of $\Pi_{1,2}$. Starting from the root of the composition tree, both the children nodes are considered and every time two accepting transcripts with the same first-round message are found, $\mathsf{Extract}_{1,2}$ is called again. The base case for the recursion is a leaf node having two accepting transcripts with the same first-round message. In this case, the extractor of the base Σ-protocol is called instead. Since the computational special soundness of $\Pi_{1,\ell}$ is proven using $\mathsf{Extract}_{1,\ell}$ shown above, we are guaranteed (except with negligible probability) that *at least* one of such leaf nodes exists. This in turn implies that at each level of the tree there is *at least* one node having two accepting transcripts with the same first-round message that we can give as input to $\mathsf{Extract}_{1,2}$. However, in principle there could be more than one of such nodes in each level. On such nodes $\mathsf{Extract}_{1,2}$ would be called again, and we would get at least one witness for the corresponding OR relation (recall that $\Pi_{1,2}$ proves the OR relation on two statements, such statements may be OR statements as well). The extraction algorithm will recursively lead to at least one witness for one base statement. Given the way the composition tree is constructed, witnesses extracted from different nodes are always related to different statements. For the sake of simplicity, let us focus on the case in which exactly one witness is extracted from each pair of accepting transcripts of $\Pi_{1,\ell}$. We now have to argue why all these k witnesses must be related to different statements. Let us assume that two witnesses related to the same statement are extracted from two executions of $\Pi_{1,\ell}$. Let us consider the composition trees of such executions. The computational special soundness of Π_{ord} guarantees (except with negligible probability) that there is one level in which, using the extractor of Π_{ord}, we extract a trapdoor for a different edge in each of the two trees (namely, a trapdoor for the first and the second parameter of the commitment scheme). If we consider such level, we can define a sub-tree containing the leaf corresponding to the extracted witness and having as root the parent node corresponding to the first edge in which we extracted a trapdoor for different edges. We are now able to break the fixed equivocation property of the commitment scheme at the first level of one of the two of such sub-trees. Indeed, consider the sub-tree having the extracted trapdoor on the same side of the extracted witness. W.l.o.g. let it be the left side. Since we extracted a witness for the left side, the first-round message of the corresponding left side Σ-protocol is the same in both transcripts, while the right-side first-round message is equivocated (otherwise, we would have extracted a witness also for this side). This means we already have a commitment which equivocates a message on the right side. Thanks to the extracted trapdoor and the partial equivocation property, we are now able to construct a fresh commitment w.r.t. the same parameters which successfully equivocates the left side. This would break the fixed equivocation property of the commitment scheme, thus reaching a contradiction. The actual proof is slightly more involved since, as we already argued above, from each pair of executions of $\Pi_{1,\ell}$ it is generally possible to extract more than one witness. Therefore, in general we are not guaranteed to find a sub-tree having a level with an extracted trapdoor on one side and an equivocated commitment on the other side. However, in the full proof we exploit the fact that the number of extracted witnesses per composition tree using $\mathsf{Extract}_{1,\ell}$ is strictly less than k[11] to argue that, among all the k pairs of composition trees with the same first-round message, there always exists a tree on which we are able to run the above reduction to the partial equivocation and the fixed equivocation properties of the commitment scheme. We prove this by induction. We exploit the fact that at each level of the tree there are *at most* two configurations of the extracted trapdoors and witnesses that do not allow the above reduction at that level. Furthermore, we use the observation that extracting a witness from a node at level $i-1$ requires extracting a witness from at least one of its children nodes at level i.

Computational special soundness full proof. Before proving that $\Pi_{\mathsf{k},\ell}$ is computational special sound we will first make some observations and introduce an additional lemma that we will use later on.

From now on, we consider the following notion of composition tree $\mathcal{T}$ for an accepting transcript of $\Pi_{1,\ell}$. Given an accepting transcript (a, c, z) of $\Pi_{1,\ell}$, for an instance $(x_1, \ldots, x_\ell)$, where a contains a 1-out-of-2 equivocal commitment to values a_{left} and a_{right}, we label the nodes of $\mathcal{T}$ as follows. Every node of $\mathcal{T}$ is inductively labeled with an instance and an accepting transcript for that instance:

- The root is labeled with $\Sigma_{\mathsf{root}} = ((x_1, \ldots, x_\ell), a, c, z)$.
- Given a node m at level q, with $q \in [\log \ell]$, labeled with $\Sigma_m = ((x_i, \ldots, x_j), a_m, c, z_m = (\tilde{z}_m, r_m, (p_0^q, p_1^q)))$, for $1 \leq i < j \leq \ell$, the

[11] All witnesses extracted from the same composition tree are related to different statements by construction. Thus, in this case we would not need to show any reduction.

left child node of m is at level $q+1$ and is labeled with $\Sigma_{\mathsf{left}} = ((x_i, \dots, x_{(j+i-1)/2}), a_{\mathsf{left}}, c, \tilde{z}_m)$, and the right child node is at level $q+1$ and is labeled with $\Sigma_{\mathsf{right}} = ((x_{((j+i+1)/2)+1}, \dots, x_j), a_{\mathsf{right}}, c, \tilde{z}_m)$[12], where a_{left} is the first-round message of the transcript for statement $(x_i, \dots, x_{(j+i-1)/2})$ and a_{right} is the first-round message of the transcript for statement $(x_{((j+i+1)/2)+1}, \dots, x_j)$.

Moreover, for each node m at level q in $\mathcal{T}$ labeled with $\Sigma_m = ((x_i, \dots, x_j), a_m, c, z_m = (\tilde{z}_m, r_m, (p_0^q, p_1^q)))$, the edge going from m to its left child is labeled with p_0^q and the edge from m to its right child is labeled with p_1^q. We say that the edges from m to their children are edges at level q.

Let (a, c, z) be an accepting transcript for $\Pi_{1,\ell}$, we notice that $\mathbf{p} = ((p_0^1, p_1^1), \dots, (p_0^{\log \ell}, p_1^{\log \ell}))$ contained in z represents the labeling of the edges of the composition tree $\mathcal{T}$ for (a, c, z). Indeed, each node m in level q, for $q \in [\log \ell]$, has the edge to the left child labeled with p_0^q and the edge to the right child labeled with p_1^q. We denote with $\mathcal{T}^{\mathbf{p}}$ a composition tree having an edge labeling defined by $\mathbf{p}$. An example of a composition tree is illustrated in Fig. 4.1.

The extractor of $\Pi_{1,\ell}$ in terms of composition trees. Let us now review how $\mathsf{Extract}_{\Pi_{1,\ell}}$, the extractor of $\Pi_{1,\ell}$ proposed in [62], works in terms of composition trees. $\mathsf{Extract}_{\Pi_{1,\ell}}$ takes in input two composition trees whose roots have accepting transcripts with the same first-round massage and different challenges. Due to stackability, we can look at every node of a composition tree as representing the transcript for the Σ-protocol for an OR relation on a subset of all the ℓ involved statements. The root of the tree Σ_{root} represents the transcript for the Σ-protocol for the whole relation $\mathcal{R}_{1,\ell}$. For each node with statements $(x_i, \dots, x_j)$, its left child Σ_{left} and right child Σ_{right} represent the Σ-protocol transcripts for a relation $\mathcal{R}_{1,(\mathsf{j}-\mathsf{i}+1)/2}$ on the first and second half of the subset of instances respectively. It follows that the leaves represent the Σ-protocols transcripts for the individual instances x_i

[12]We recall that $\tilde{z}_m$ contains the commitment parameters used to generate the first-round message of the children of node m. Therefore, all the children of node m use the same commitment parameters pair.

with $i \in [\ell]$. $\mathsf{Extract}_{\Pi_{1,\ell}}$ is a recursive composition of the extractor $\mathsf{Extract}_{\Pi_{1,2}}$ of the underlying $\Pi_{1,2}$.

Consider the following two accepting transcripts Σ^1_{root} and Σ^2_{root} (with associated composition trees $\mathcal{T}^1$ and $\mathcal{T}^2$) having the same first-round message $a_{\mathsf{root}} = (\mathtt{com}, p_0, p_1)$, $c^1 \neq c^2$, $z^1 = (\tilde{z}^1, r, p_0, p_1)$, $z^2 = (\tilde{z}^2, r, p_0, p_1)$. Given these accepting transcripts, it is possible to compute transcripts for Σ_{left} and Σ_{right} as follows: $a^1_{\mathsf{left}} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{left}}(x_{\mathsf{left}}, c^1, \tilde{z}^1)$, $a^1_{\mathsf{right}} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{right}}(x_{\mathsf{right}}, c^1, \tilde{z}^1)$, $a^2_{\mathsf{left}} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{left}}(x_{\mathsf{left}}, c^2, \tilde{z}^2)$, $a^2_{\mathsf{right}} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{right}}(x_{\mathsf{right}}, c^2, \tilde{z}^2)$, where $x_{\mathsf{left}} = (x_1, \ldots, x_{\ell/2})$ and $x_{\mathsf{right}} = (x_{\ell/2+1}, \ldots, x_\ell)$. Therefore, whenever $a^1_{\mathsf{left}} = a^2_{\mathsf{left}}$ or $a^1_{\mathsf{right}} = a^2_{\mathsf{right}}$ it is possible to call again $\mathsf{Extract}_{\Pi_{1,2}}$ on the two composition sub-trees of $\mathcal{T}^1$ and $\mathcal{T}^2$ rooted at the respective children nodes that have the same-first round messages (i.e., either the left or the right children nodes). For each pair of nodes in $\mathcal{T}^1$ and $\mathcal{T}^2$ having the same first-round message, looking again at the children nodes, either a^1_{left} is equal to a^2_{left} or a^1_{right} is equal to a^2_{right} with overwhelming probability, otherwise it is possible to break the computational special soundness of $\Pi_{1,2}$ and thus the computational special soundness of $\Pi_{1,\ell}$. Therefore, the latter leads to at least one extracted witness for an instance x_j with $j \in [\ell]$. In general, up to ℓ witnesses could be extracted.

Trapdoor equivalence class. We now introduce the concept of trapdoor equivalence class. Informally, a trapdoor equivalence class identifies all vectors of trapdoors having the same trapdoor (and non-trapdoor) position for all parameters pairs.

Definition 22 (Trapdoor Equivalence Class). *Let* $\mathbf{p} = ((p^1_0, p^1_1), \ldots, (p^n_0, p^n_1))$ *be a vector of parameters of a 1-out-of-2 equivocal commitment scheme (cfr., Sec. 4.1.4.2), and let* $\mathcal{R}_T$ *be its associated poly-time relation. Let* $\mathbf{td} = (\mathsf{td}_1, \ldots, \mathsf{td}_n)$ *be a vector of trapdoors such that for every* $i \in [n]$*, there exists* $\beta \in \{0,1\}$ *such that* $\mathcal{R}_T(p^i_\beta, \mathsf{td}_i) = 1$*. The trapdoor equivalence class* $[\mathbf{td}]^{\mathcal{R}_T}_{\mathbf{p}}$ *is the set containing all vectors* $\mathbf{td}' = (\mathsf{td}'_1, \ldots, \mathsf{td}'_n)$ *in which for every* $i \in [n]$*, and* $\beta \in \{0,1\}$*,* $\mathcal{R}_T(p^i_\beta, \mathsf{td}_i) = \mathcal{R}_T(p^i_\beta, \mathsf{td}'_i)$*.*

We now define the notion of *valid* trapdoor equivalence class. Informally, a trapdoor equivalence class is said to be valid w.r.t. two composition trees with the same edge labels if there is no level in such trees where a trapdoor is related to a parameter (e.g., in the first position) while an equivocation is performed in the position of the other parameter (e.g., the second position).

Definition 23 (Valid Trapdoor Equivalence Classes)**.** *Let* $\mathbf{p} = ((p_0^1, p_1^1), \dots, (p_0^{\log \ell}, p_1^{\log \ell}))$ *be a vector of parameters of a 1-out-of-2 equivocal commitment scheme (cfr., Sec. 4.1.4.2) with associated poly-time relation* $\mathcal{R}_T$*. Let* (a, c, z) *and* (a, c', z')*, with* $c \neq c'$*, be two accepting transcripts of* $\Pi_{1,\ell}$ *for the same instance* $x = (x_1, \dots, x_\ell)$*, and* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$ *be the composition trees associated with those accepting transcripts. Let us take* $(\mathsf{td}_1, \dots, \mathsf{td}_{\log \ell}) \in [\mathbf{td}]_{\mathbf{p}}^{\mathcal{R}_T}$*. A trapdoor equivalence class* $[\mathbf{td}]_{\mathbf{p}}^{\mathcal{R}_T}$ *is valid w.r.t.* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$ *if, for every level* $i \in [\log \ell]$ *(of both* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$*), there does not exist a node* m *at level* i *in which*

$$((\mathcal{R}_T(p_0^i, \mathsf{td}_i) = 1) \wedge (a_{\mathsf{right}}^{(i,m)} \neq a'^{(i,m)}_{\mathsf{right}})) \vee ((\mathcal{R}_T(p_1^i, \mathsf{td}_i) = 1) \wedge (a_{\mathsf{left}}^{(i,m)} \neq a'^{(i,m)}_{\mathsf{left}}))$$

where $a_{\mathsf{left}}^{(i,m)}$ *(resp.* $a_{\mathsf{right}}^{(i,m)}$*) is the first-round message associated to the left (resp. right) child of node* m *which is at level* i *of* $\mathcal{T}^{\mathbf{p}}$*. The same holds w.r.t* $a'^{(i,m)}_{\mathsf{left}}$*,* $a'^{(i,m)}_{\mathsf{right}}$*, and* $\mathcal{T}'^{\mathbf{p}}$*.*

Lemma 2. *Let* (a, c, z) *and* (a, c', z') *be two accepting transcripts for* $\Pi_{1,\ell}$ *for the same statement* $(x_1, \dots, x_\ell)$ *and* $c \neq c'$*. Let* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$ *be the two composition trees associated with* (a, c, z) *and* (a, c', z') *respectively. Let* CS *be the 1-out-of-2 equivocal commitment scheme with associated relation* $\mathcal{R}_T$ *used in* $\Pi_{1,\ell}$*. If* $\mathsf{Extract}_{\Pi_{1,\ell}}$ *extracts* s *different witnesses, then the number of valid trapdoor equivalence classes* $[\mathbf{td}]_{\mathbf{p}}^{\mathcal{R}_T}$ *w.r.t.* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$ *is upper-bounded by* s*.*

Proof. Let i be a level of $\mathcal{T}^{\mathbf{p}}$ and $\mathcal{T}'^{\mathbf{p}}$ and t be the number of *valid* trapdoor equivalence classes $[\mathbf{td}]_{\mathbf{p}}^{\mathcal{R}_T}$ (see Def. 23). Let us call $\mathbf{td}^j = (\mathsf{td}_1^j, \dots, \mathsf{td}_{\log \ell}^j)$ the representative of the j-th valid trapdoor equivalence class $[\mathbf{td}^j]_{\mathbf{p}}^{\mathcal{R}_T}$ for $j \in [t]$. We define L_i, E_i and s_i as follows:

- $L_i = \left\lceil \frac{\sum_{j \in [t]} (\mathcal{R}_T(p_0^i, \mathsf{td}_i^j))}{t} \right\rceil + \left\lceil \frac{\sum_{j \in [t]} (\mathcal{R}_T(p_1^i, \mathsf{td}_i^j))}{t} \right\rceil$. L_i can either be 1 or 2. L_i is 1 if, at level i, either $\mathcal{R}_T(p_0^i, \mathsf{td}_i^j) = 1$ and $\mathcal{R}_T(p_1^i, \mathsf{td}_i^k) = 0$ or $\mathcal{R}_T(p_0^i, \mathsf{td}_i^k) = 0$ and $\mathcal{R}_T(p_1^i, \mathsf{td}_i^j) = 1$ for some $j, k \in [t]$. Otherwise, L_i is 2.

- $E_i = \prod_{j=1}^{i} L_j$ E_i represents the number of valid equivalence classes for the sub-trees of $\mathcal{T}^{\mathbf{p}}$ and $\mathcal{T}'^{\mathbf{p}}$ having only the first i levels. $E_{\log \ell}$ is equal to t.
- s_i represents the number of witnesses that can be extracted from the i-th level of $\mathcal{T}^{\mathbf{p}}$ and $\mathcal{T}'^{\mathbf{p}}$. It is trivial to see that $s_i \geq s_{i-1}$ for every $i \in [\log \ell]$.

In the following, we use $a^{(i,m)}$ to indicate the first-round message contained in the node m at level i in composition tree $\mathcal{T}^{\mathbf{p}}$. Additionally, $a^{(i,m)}_{\mathsf{left}}$ indicates the first-round message contained in the left child of node m at level i of composition tree $\mathcal{T}^{\mathbf{p}}$. The pedix "right" indicates a right child node. The apex " $'$ " is added to indicate that a node is from composition tree $\mathcal{T}'^{\mathbf{p}}$.

We prove the lemma by induction. For every level $i \in [\log \ell]$, $E_i \leq s_i$.

Base case: Let us consider the root m of $\mathcal{T}^{\mathbf{p}}$ and $\mathcal{T}'^{\mathbf{p}}$. By inspection, it is clear that if $s_0 = 1$ then either $a^{(0,m)}_{\mathsf{left}} = a'^{(0,m)}_{\mathsf{left}}$ or $a^{(0,m)}_{\mathsf{right}} = a'^{(0,m)}_{\mathsf{right}}$ but not both at the same time. Indeed, in this last case s_0 would be equal to 2. Therefore, fixed one of the two cases above, there exists only one *valid* equivalence class, thus $L_0 = 1$. Let us assume that $a^{(0,m)}_{\mathsf{left}} = a'^{(0,m)}_{\mathsf{left}}$ and $a^{(0,m)}_{\mathsf{right}} \neq a'^{(0,m)}_{\mathsf{right}}$. Indeed, if $L_0 = 2$ and $s_0 = 1$, then there exists td^j_1 and td^k_1 from two different equivalence classes with representative td^j and td^k respectively such that either $\mathcal{R}_T(p_0, \mathsf{td}^j_1) = 1$ or $\mathcal{R}_T(p_0, \mathsf{td}^k_1) = 1$ violating the validity property of Def. 23. If $s_1 = 2$ the base case is trivially true since $L_1 \leq 2$. Therefore, in the base case it holds that $E_2 = L_2 \leq s_2$.

Inductive case: We now show that $E_{i-1} \leq s_{i-1}$ implies $E_i \leq s_i$. We notice that for increasing values of i the values of E_i and s_i are monotone non-decreasing. Indeed, E_i is a product of non-zero integers. Regarding s_i, extracting a witness from a node at level $i-1$ requires extracting a witness from at least one of its child nodes at level i. Therefore, if $E_i = E_{i-1}$ then $E_i \leq s_i$ trivially follows from $E_{i-1} \leq s_{i-1}$. We now show that if $E_i = 2E_{i-1}$, then

$s_i \geq 2s_{i-1}$. If $E_i = 2E_{i-1}$ then it holds that $L_{i-1} = 2$ which means that there exists td_i^j and td_i^k in equivalence classes with representatives td^j and td^k satisfying either $\mathcal{R}_T(p_0^i, \mathsf{td}_i^j) = 1$ and $\mathcal{R}_T(p_1^i, \mathsf{td}_i^k) = 1$ or $\mathcal{R}_T(p_0^i, \mathsf{td}_i^k) = 1$ and $\mathcal{R}_T(p_1^i, \mathsf{td}_i^j) = 1$. Let us assume by contradiction that $s_i < 2s_{i-1}$. In this case there must be at least a witness at level $i-1$ such that $a_{\mathsf{left}}^{(i,m)} = a'^{(i,m)}_{\mathsf{left}}$ and $a_{\mathsf{right}}^{(i,m)} \neq a'^{(i,m)}_{\mathsf{right}}$ or $a_{\mathsf{right}}^{(i,m)} = a'^{(i,m)}_{\mathsf{right}}$ and $a_{\mathsf{left}}^{(i,m)} \neq a'^{(i,m)}_{\mathsf{left}}$ for some node m. From the above observation, it follows that at least one of the trapdoor equivalence classes does not satisfy the validity requirement of Def. 23. Then, it holds that $s_i \geq 2s_{i-1}$, from which it follows that $E_i \leq s_i$.

□

Lemma 3 (Computational Special Soundness). *$\Pi_{\mathsf{k},\ell}$ is computational special sound.*

Proof. Computational special soundness follows from the computational special soundness of $\Pi_{1,\ell}$, the computational special soundness of Π_{ord}, and the fixed equivocation property of CS. To prove computational special soundness of $\Pi_{\mathsf{k},\ell}$ we define the extractor $\mathsf{Extract}_{\Pi_{\mathsf{k},\ell}}$ based on the extractor $\mathsf{Extract}_{\Pi_{1,\ell}}$ of $\Pi_{1,\ell}$ and the extractor $\mathsf{Extract}_{\Pi_{ord}}$ of Π_{ord} as follows.

Given two accepting transcripts with the same first-round message $a = (a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$, $c \neq c'$, $z = (z_1, \ldots, z_k, z_{ord})$, and $z' = (z'_1, \ldots, z'_k, z'_{ord})$ for statement $(x_1, \ldots, x_\ell)$, $\mathsf{Extract}_{\Pi_{\mathsf{k},\ell}}$ runs, for each $i \in [k]$, the extractor of each $(1,\ell)$-PTR, i.e. $\mathsf{Extract}_{\Pi_{1,\ell}}((x_1 \ldots, x_\ell), a_i, c, c', z_i, z'_i)$.

From each execution of $\mathsf{Extract}_{\Pi_{1,\ell}}((x_1 \ldots, x_\ell), a_i, c, c', z_i, z'_i)$, $\mathsf{Extract}_{\Pi_{\mathsf{k},\ell}}$ obtains with overwhelming probability at least a witness (w_1^i, j_1^i), otherwise it is possible to break the computational special soundness of $\Pi_{1,\ell}$. Indeed, (a_i, c, z_i) and (a_i, c', z'_i) are two accepting transcripts for statement $(x_1 \ldots, x_\ell)$ for $\Pi_{1,\ell}$ otherwise $\mathsf{V}^{\mathsf{k},\ell}$ cannot accept the transcripts (a, c, z) and (a, c', z'). If $\mathsf{Extract}_{\Pi_{1,\ell}}((x_1 \ldots, x_\ell), a_i, c, c', z_i, z'_i)$ does not return a valid witness, then $\mathsf{ExpExt}_{\mathsf{P}1,\ell*,\mathsf{Extract}_{\Pi_{1,\ell}}}(x_1, \ldots, x_\ell)$ returns 1, that happens only with negligible probability.

Let us consider the case in which, for all $i \in [k]$, a set of witnesses $\mathbf{w}^i = \{(w_1^i, j_1^i), \ldots, (w_k^i, j_{fi}^i)\}$ with $f^i < k$ is extracted from each $(1,\ell)$-

PTR. Recall that for $u \in [f^i]$, each index $j^i_u \in [\ell]$ simply specifies the base instance the extracted witness corresponds to. Additionally, consider $I^i = \{j^i_1, \dots, j^i_{f^i}\}$ and the case for which $|\bigcup_{i=1}^k I^i| < k$. Namely, this is the case in which less than k witnesses for different statements in $(x_1, \dots, x_\ell)$ are extracted from the (k, ℓ)-PTR. We only need to consider this case since otherwise we would have already extracted a witness for relation $\mathcal{R}_{\mathsf{k},\ell}$. This case implies that from each $(1, \ell)$-PTR less than k witnesses are extracted, since the witness extracted by $\mathsf{Extract}_{\Pi_{1,\ell}}$ are always related to different elementary statements by construction.

We run $\mathsf{Extract}_{\Pi_{ord}}$ on input $((\mathbf{p}_i)_{i\in[k]}, a_{ord}, c, c', z_{ord}, z'_{ord})$ obtaining, with overwhelming probability, a witness for $(\mathbf{p}_i)_{i\in[k]}$ being in $\mathcal{R}_{ord}$. Otherwise, it is possible to break the computational special soundness of Π_{ord}. The reduction follows the same blueprint of the one shown for the computational special soundness of $\Pi_{1,\ell}$. We now argue that there must be two composition trees $\mathcal{T}^{\mathbf{p}_i}$ and $\mathcal{T}'^{\mathbf{p}_i}$, with $i \in [k]$, in which there exists a node m at level $q \in [\log \ell]$ in which $\mathcal{R}_T(p^q_0, \mathsf{td}^i_q) = 1$, $a^{(q,m)}_{\mathsf{left}} = a'^{(q,m)}_{\mathsf{left}}$ and $a^{(q,m)}_{\mathsf{right}} \neq a'^{(q,m)}_{\mathsf{right}}$ (or equally $\mathcal{R}_T(p^q_1, \mathsf{td}^i_q) = 1$, $a^{(q,m)}_{\mathsf{right}} = a'^{(q,m)}_{\mathsf{right}}$, and $a^{(q,m)}_{\mathsf{left}} \neq a'^{(q,m)}_{\mathsf{left}_q}$), where td^i_q is an element of the vector $\mathbf{td}^i = (\mathsf{td}^i_1, \dots, \mathsf{td}^i_{\log \ell})$ related to $\mathbf{p}_i$, and $\mathbf{td}^i$ was extracted using $\mathsf{Extract}_{\Pi_{ord}}$. Namely, $\mathbf{td}^i$ is a representative of a trapdoor equivalence class which is not valid according to Def. 23. Indeed, by Lemma 2 the number of valid trapdoor equivalence classes is upper-bounded by the number of extracted witnesses, which is strictly less than k in this case. Nevertheless, thanks to the computational special soundness of Π_{ord}, $\mathsf{Extract}_{\Pi_{\mathsf{k},\ell}}$ extracts, with overwhelming probability, k vectors of trapdoors, all belonging to different trapdoor equivalence classes[13]. Therefore, one of such vectors must belong to a non-valid trapdoor equivalence class, thus allowing the following reduction. Consider the transcripts associated to node m at level q in both $\mathcal{T}^{\mathbf{p}_i}$ and $\mathcal{T}'^{\mathbf{p}_i}$. They are of the form $(a^{(q,m)}, c, z), (a'^{(q,m)}, c', z')$ with $a^{(q,m)} = a'^{(q,m)}$, $c \neq c'$, $z = (z^*, r, p^q_0, p^q_1)$, and $z' = (z^{*\prime}, r', p^q_0, p^q_1)$. We now use these two accepting transcripts to break the fixed equivocation property of the 1-out-of-2 equivocal commitment scheme. W.l.o.g. we consider the case $a^{(q,m)}_{\mathsf{left}} = a'^{(q,m)}_{\mathsf{left}}$ and $a^{(q,m)}_{\mathsf{right}} \neq a'^{(q,m)}_{\mathsf{right}}$. $\mathcal{A}$ does the following:

[13]Indeed, this is the requirement imposed by the relation $\mathcal{R}_{ord}$.

- $(m_0, m'_0, m_1) \leftarrow_\$ \{0,1\}^{3\lambda}$ with $m_0 \neq m'_0$.
- $(\mathtt{com}', \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta = 1, m_1, p_0^q, p_1^q, \mathsf{td}_q^i)$;
- $r^* \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta = 1, m_0, m_1, p_0^q, p_1^q, \mathsf{td}_q^i, \mathsf{aux})$
- $r^{*\prime} \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta = 1, m'_0, m_1, p_0^q, p_1^q, \mathsf{td}_q^j, \mathsf{aux})$

$\mathcal{A}$ outputs $(p_0^q, p_1^q, r, r', r^*, r^{*\prime}, a_{\mathsf{left}}^{(q,m)}, a_{\mathsf{left}}^{\prime(q,m)}, a_{\mathsf{right}}^{(q,m)}, a_{\mathsf{right}}^{\prime(q,m)}, m_0, m'_0, m_1, m_1)$.

Thanks to the partial equivocation property of the commitment scheme, $\mathsf{BindCom}(\mathsf{pp}, p_0^q, p_1^q, m_0, m_1, r^*) = \mathsf{BindCom}(\mathsf{pp}, p_0^q, p_1^q, m_0, m'_1, r^{*\prime})$. Additionally, since the two transcripts are accepting, we have that $\mathsf{BindCom}(\mathsf{pp}, p_0^q, p_1^q, a_{\mathsf{left}}^{(q,m)}, a_{\mathsf{right}}^{(q,m)}, r) = \mathsf{BindCom}(\mathsf{pp}, p_0^q, p_1^q, a_{\mathsf{left}}^{\prime(q,m)}, a_{\mathsf{right}}^{\prime(q,m)}, r')$. Therefore, $\mathcal{A}$ breaks the fixed equivocation property with the same probability that less than k witnesses for different elementary statements are extracted, thus reaching a contradiction. $\square$

For the sake of clarity, we first report a sketch of the proof of the EHVZK property. Then, we report a detailed proof of the theorem.

Extended honest-verifier zero knowledge proof skech. We name the EHVZK simulator of $\Pi_{\mathsf{k},\ell}$ as $\mathcal{S}_{\mathsf{k},\ell}$ and we use $D_{\mathsf{k},\ell}$ to indicate the third-round messages distribution of $\Pi_{\mathsf{k},\ell}$. $\mathcal{S}_{\mathsf{k},\ell}$, on input statement $(x_1, \dots, x_\ell)$, challenge c, and third-round message $z = (z_1, \dots, z_k, z_{ord})$ sampled from $D_{\mathsf{k},\ell}$, outputs the first-round message $(a_1, \dots, a_k, a_{ord}, \mathbf{p}_1, \dots, \mathbf{p}_k)$. Recall that the EHVZK property requires that all third-round messages of honest protocol executions for statements $x \in \mathcal{L}$ follow the same distribution. Such distribution must be efficiently samplable. Furthermore, running the simulator on input statement $x \in \mathcal{L}$, a uniformly random challenge c, and a third-round message z sampled from such distribution, deterministically produces a first-round message a so that (a, c, z) is indistinguishable from honest protocol execution transcripts (cfr., Sec. 4.1.4.1). Notice that sampling from $D_{\mathsf{k},\ell}$ simply consists of sampling each z_i, with $i \in [k]$, from $D_{1,\ell}$ (i.e., the third-round messages distribution of $\Pi_{1,\ell}$), and sampling z_{ord} from D_{ord} (i.e, the third-round messages distribution of Π_{ord}). Since both $\Pi_{1,\ell}$ and Π_{ord} are EHVZK, such distributions exist and are efficiently samplable.

We construct $\mathcal{S}_{\mathsf{k},\ell}$ in terms of the EHVZK simulators of the underlying protocols. We name the EHVZK simulator of $\Pi_{1,\ell}$ as $\mathcal{S}_{1,\ell}$ and we use $\mathcal{S}_{ord}$ to indicate the EHVZK simulator of Π_{ord}. $\mathcal{S}_{\mathsf{k},\ell}$ parses each z_j, with $j \in [k]$, and selects the commitment parameters $\mathbf{p}_j = ((p_0^{(1,j)}, p_1^{(1,j)}), \ldots, (p_0^{(\log \ell, j)}, p_1^{(\log \ell, j)}))$. $\mathcal{S}_{\mathsf{k},\ell}$ gives in input to $\mathcal{S}_{ord}$ the instance $(\mathbf{p}_1, \ldots, \mathbf{p}_k)$, the challenge c, and z_{ord}, thus obtaining a_{ord}. Then, $\mathcal{S}_{\mathsf{k},\ell}$ runs $\mathcal{S}_{1,\ell}((x_1, \ldots, x_\ell), c, z_i)$ for each of the k $(1,\ell)$-PTR obtaining $(a_1, \ldots, a_k)$. Finally, $\mathcal{S}_{\mathsf{k},\ell}$ outputs $(a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$. We now show that $\Pi_{\mathsf{k},\ell}$ is EHVZK via hybrid arguments. The first hybrid $\mathcal{H}_1$ corresponds to the real protocol execution, except that z_{ord} is sampled from D_{ord} and a_{ord} is obtained running $\mathcal{S}_{ord}$. The real protocol and $\mathcal{H}_1$ are indistinguishable thanks to the EHVZK of Π_{ord}. Then, a sequence of k hybrids $\mathcal{H}_2, \ldots, \mathcal{H}_{k+1}$ follows; each hybrid $\mathcal{H}_i$, with $i \in \{2, \ldots, k+1\}$ differs from the previous one because z_i, the third-round message of the i-th $(1,\ell)$-PTR, is sampled from $D_{1,\ell}$, and the first-round message a_i is computed using $\mathcal{S}_{1,\ell}$. Each hybrid is indistinguishable from its predecessor due to the EHVZK of $\Pi_{1,\ell}$. The proof of EHVZK of $\Pi_{\mathsf{k},\ell}$ ends by observing that $\mathcal{H}_{k+1}$ is identical to $\mathcal{S}_{\mathsf{k},\ell}$. Notice that Π_{ord} is perfect EHVZK and that $\Pi_{1,\ell}$ preserves the EHVZK flavour of the composed protocols. Thus, $\Pi_{\mathsf{k},\ell}$ clearly preserves the EHVZK flavour of the composed protocols.

Lemma 4 (Extended Honest-Verifier Zero Knowledge). $\Pi_{\mathsf{k},\ell}$ *is Extended Honest-Verifier Zero Knowledge.*

Proof. Let $D_{x,c}^{(z)^*}$ be the third-round message distribution of $\Pi_{1,\ell}$, and $D_c^{(z)'}$ be the third-round message distribution of Π_{ord}. Let $\mathcal{S}_{1,\ell}^{\mathsf{EHVZK}}$ be the EHVZK simulator of $\Pi_{1,\ell}$, for $i \in [k]$, and $\mathcal{S}_{ord}^{\mathsf{EHVZK}}$ be the EHVZK simulator of Π_{ord}. Let $D_c^{(z)} = \Big\{(z_1, \ldots, z_k, z_{ord}) | \, \forall i \in [k] \; z_i \leftarrow_\$ D_c^{(z)^*}, z_{ord} \leftarrow_\$ D_c^{(z)'}\Big\}$.

We define simulator $\mathcal{S}_{\mathsf{k},\ell}^{\mathsf{EHVZK}}(\mathbf{x} = (x_1, \ldots, x_\ell), c, (z_1, \ldots, z_k, z_{ord}))$ as follows:

1. Compute $a_i \leftarrow \mathcal{S}_{1,\ell}^{\mathsf{EHVZK}}(\mathbf{x}, c, z_i)$, for all $i \in [k]$;
2. Parse $z_i = (\tilde{z}_j, \{r_j\}_{j \in [\log \ell]}, \mathbf{p}_i)$, for all $i \in [k]$;
3. Compute $a_{ord} \leftarrow \mathcal{S}_{ord}^{\mathsf{EHVZK}}((\mathbf{p}_1, \ldots, \mathbf{p}_k), c, z_{ord})$;

4. Return $(a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$.

We prove that $\Pi_{\mathsf{k},\ell}$ is EHVZK with the following hybrid arguments.

$\mathcal{H}_0$: this is equal to the real game with honest prover, except that the prover of hybrid $\mathcal{H}_0$ takes in input $(\mathbf{x}, \mathbf{w}, c, (z_1, \ldots, z_k, z_{ord}))$, where $z_i \in D_c^{(z)}$ for all $i \in [k]$ and $z_{ord} \in D_c^{(z)'}$. The additional inputs c and $(z_1, \ldots, z_k, z_{ord})$ are ignored during the execution by the prover of hybrid $\mathcal{H}_0$. We notice that $\mathcal{H}_0$ is distributed identically to the real game.

$\mathcal{H}_{1_0}$: It is identical to $\mathcal{H}_0$ except that a_{ord} is computed using $\mathcal{S}_{ord}^{\mathsf{EHVZK}}((\mathbf{p}_1, \ldots, \mathbf{p}_k), c, z_{ord})$ where $(\mathbf{p}_1, \ldots, \mathbf{p}_k)$, c and z_{ord} are taken from the prover's additional input specified in $\mathcal{H}_0$. Recall that z_i, for $i \in [k]$, contains also $\mathbf{p}_i$. If there exists a distinguisher D_{EHVZK} that distinguishes $\mathcal{H}_0$ from $\mathcal{H}_{1_0}$ with non-negligible advantage, we can construct a distinguisher D_{ord} that distinguishes an execution of $\mathcal{S}_{ord}^{\mathsf{EHVZK}}$ from an execution of Π_{ord} with the same advantage. The reduction works as follows:

- D_{ord} samples randomness $\mathtt{rand}$, computes $(a_j, \mathbf{td}_j) \leftarrow \mathsf{P}_0^{1,\ell}(\mathbf{x}, (w_j, i_j); \mathtt{rand})$ and $z_j = (\tilde{z}_j, \{r_i\}_{i \in [\log \ell]}, \mathbf{p}_j) \leftarrow \mathsf{P}_1^{1,\ell}(\mathbf{x}, (w_j, i_j), c; \mathtt{rand})$, for all $j \in [k]$. Then, D_{ord} outputs the statement/witness pair $((\mathbf{p}_1, \ldots, \mathbf{p}_k), (\mathbf{td}_1, \ldots, \mathbf{td}_k))$ in the experiment (cfr., Def. 32) $\mathsf{ExpEHVZK}_{\mathsf{P}',\mathsf{D}_{ord}}(c)$ (where P' is either $(\mathsf{P}_0^{ord}, \mathsf{P}_1^{ord})$ or $\mathcal{S}_{ord}^{\mathsf{EHVZK}}$) receiving back (a_{ord}, z_{ord}).
- D_{ord}, on input $((\mathbf{p}_1, \ldots, \mathbf{p}_k), (\mathbf{td}_1, \ldots, \mathbf{td}_k), a_{ord}, c, z_{ord})$, forwards $(\mathbf{x}, \mathbf{w}, a, c, z)$ to D_{EHVZK}, where $a = (a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$ and $z = (z_1, \ldots, z_k, z_{ord})$. D_{ord} outputs whatever D_{EHVZK} outputs.

$\mathcal{H}_{1_i}$: for each $i \in [k]$, this is equal to $\mathcal{H}_{1_{i-1}}$ except that all values $a_1, \ldots, a_i$ are computed using $\mathcal{S}_{1,\ell}^{\mathsf{EHVZK}}(\mathbf{x}, c, z_i)$ where c and z_i are taken from the prover's additional input specified in $\mathcal{H}_0$.

If there exists a distinguisher D_{EHVZK} that distinguishes $\mathcal{H}_{1_i}$ from $\mathcal{H}_{1_{i+1}}$ with non-negligible advantage, we can construct a distinguisher $\mathsf{D}_{1,\ell}$ that distinguishes a simulated execution $\mathcal{S}_{1,\ell}^{\mathsf{EHVZK}}$

from an execution of $\Pi_{1,\ell}$ with the same advantage. The reduction works as follows:

- $\mathsf{D}_{1,\ell}$ samples randomness $\mathtt{rand}$, $\mathsf{D}_{1,\ell}$ computes $(a_j, \mathbf{td}_j) \leftarrow \mathsf{P}_0^{1,\ell}(\mathbf{x}, (w_j, i_j); \mathtt{rand})$ and $z_j = (\tilde{z}_j, \{r_i\}_{i \in [\log \ell]}, \mathbf{p}_j) \leftarrow \mathsf{P}_1^{1,\ell}(\mathbf{x}, (w_j, i_j), c; \mathtt{rand})$, for each $j < i$. Then, D_{ord} samples $z_j \leftarrow\$ D_c^z$ and computes $a_j \leftarrow \mathcal{S}_{1,\ell}^{\mathsf{EHVZK}}(\mathbf{x}, c, z_j)$ for each $j > i$. The value a_{ord} is computed from $\mathcal{S}^{\mathsf{EHVZK}}((\mathbf{p}_1, \dots, \mathbf{p}_k), c, z_{ord})$ for $z_{ord} \leftarrow\$ D_c^{(z)'}$. Then $\mathsf{D}_{1,\ell}$, outputs the statement/witness pair $(\mathbf{x}, w_i)$ in the experiment $\mathsf{ExpEHVZK}_{\mathsf{P}',\mathsf{D}_{1,\ell}}(c)$ (where P' is either $(\mathsf{P}_0^{1,\ell}, \mathsf{P}_1^{1,\ell})$ or $\mathcal{S}_{1,\ell}^{\mathsf{EHVZK}}$), receiving back (a_i, z_i).
- $\mathsf{D}_{1,\ell}$, on input $(\mathbf{x}, w_i, a_i, c, z_i)$, sends $(\mathbf{x}, \mathbf{w}, a, c, z)$ to D_{EHVZK}, where $a = (a_1, \dots, a_k, a_{ord}, \mathbf{p}_1, \dots, \mathbf{p}_k)$ and $z = (z_1, \dots, z_k, z_{ord})$. $\mathsf{D}_{1,\ell}$ outputs whatever D_{EHVZK} outputs.

$\mathcal{H}_2$**:** this is equal to $\mathcal{S}_{\mathsf{k},\ell}^{\mathsf{EHVZK}}(\mathbf{x}, c, (z_1, \dots, z_k, z_{ord}))$. $\mathcal{H}_2$ is identical to $\mathcal{H}_{1_k}$.

□

On the communication complexity of our (k, ℓ)-PTR. Π_{ord} has a communication complexity of $\mathcal{O}(k\lambda \log \ell)$. The communication complexity of a single execution of $\Pi_{1,\ell}$ is $\mathcal{O}(\lambda \log \ell + \mathsf{CC}(\Sigma_{base}))$, where Σ_{base} is the underlying protocol. The k repetitions of $\Pi_{1,\ell}$ lead to a communication complexity of $\mathcal{O}(k(\lambda \log \ell + \mathsf{CC}(\Sigma_{base}))$ in total. Therefore, the communication complexity of $\Pi_{\mathsf{k},\ell}$ is:
$\mathcal{O}(k(\lambda \log \ell + \mathsf{CC}(\Sigma_{base}) + k(\lambda \log \ell)) = \mathcal{O}(k(\lambda \log \ell + \mathsf{CC}(\Sigma_{base}))$.

4.1.5 Threshold Ring Signatures

A natural application of proofs over threshold relations is ring signatures. Given a set of ℓ signers, a ring signature is a digital signature that guarantees that a message was signed by a signer in the set (also called the ring) but at the same time protects the identity of the signer. There is a direct link between $(1, \ell)$-PTRs and ring signatures. For example, as pointed out in [62] one can get a ring signature whose size is logarithmic in the size of the ring by making their $(1, \ell)$-PTR non-interactive using a random oracle (see [62] Page 4 and Sec. 9.3). The ℓ public keys would be the statement of the proof, and the composed Σ-protocol would be a proof of knowledge of at least one signing key (i.e., the witness). Properly using the random oracle makes the signing process non-interactive and ties the proof to the signed message. Threshold ring signatures are a natural extension of ring signatures. In a threshold ring signature, k signers cooperate to sign a message while hiding their identity within the larger group of size ℓ. By using our construction for (k, ℓ)-PTR, one can easily get a threshold ring signature.

Let the statement $x = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ be a tuple containing the signers' verification keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$. Consider the associated $(1, \ell)$-PTR to prove knowledge of at least one signing key sk_i related to one of the verification keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$, and the (k, ℓ)-PTR to prove knowledge of at least k signing keys related to k of the ℓ verification keys. Let Σ_{base} be the base Σ-protocol to prove the knowledge of the signing key sk_i related to the verification key vk_i. Finally, let the aggregator A be a third party who aggregates the work of the k signers to create the final signature. A is trusted for anonymity, but not for unforgeability (roughly, A knows the identities of the signers, but should not be able to produce a threshold ring signature on its own, even after having performed several aggregations).

Let $\mathcal{K}$ be a set of indexes indicating the k signers within the ring of size ℓ. Consider a message m to be signed which is known to all signers and to A. Our threshold ring signature works as follows.

1. For all $i \in \mathcal{K}$, each signer S_i computes the first-round message a_i of Σ_{base} for statement vk_i and witness sk_i, and sends a_i to A.

2. Using the first-round messages for Σ_{base} received in the previous step, A computes the first-round messages $a_i^{1,\ell} = (\mathtt{com}, (p_0^j, p_1^j)_{j \in [\log \ell]})$[14] for each of the k underlying $(1, \ell)$-PTR of [62] for statement x. Moreover, A computes the first-round message a_T of the proof

[14]In [62] $a_i^{1,\ell}$ is obtained from recursively composing $\ell/2$ instances of a 1-out-of-2 equivocal commitment. Therefore, $a_i^{1,\ell}$ is of the form $(\mathtt{com}, (p_0^j, p_1^j)_{j \in [\log \ell]})$, where $(p_0^j, p_1^j)_{j \in [\log \ell]}$ is a $\log \ell$ size tuple of pair of commitment parameters.

of parameters ordering Π_{ord} with the k tuples of commitment parameters pairs $(p_0^j, p_1^j)_{j\in[\log \ell]}$ (i.e., one for each $a_i^{1,\ell}$ computed by A) as statement[15].

A sends all the first-round messages of the underlying $(1,\ell)$-PTR, (i.e., $(a_1^{1,\ell}, \ldots, a_k^{1,\ell})$), together with Π_{ord}'s first-round message a_T to each S_i.

3. S_i computes the challenge c for our (k,ℓ)-PTR compiler by hashing the statement x together with the (k,ℓ)-PTR first-round message $a = (a_1^{1,\ell}, \ldots, a_k^{1,\ell}, a_T)$ and the message m to be signed (i.e., $c = \mathsf{H}(x||a||m)$). Then, S_i computes the third-round message z_i of Σ_{base} from its first-round message a_i and the challenge c, and sends z_i to A.

4. A computes each third-round message $z_i^{1,\ell}$ of the $(1,\ell)$-PTR of [62] from the statement x, a_i, $c = \mathsf{H}(x||a||m)$, and the values z_i received from each S_i. Then, A computes the third-round message z_T of the proof of parameters ordering Π_{ord} on the same challenge c and first-round message a_T. A publishes the (k,ℓ)-threshold ring signature $\sigma = (a = (a_1^{1,\ell}, \ldots, a_k^{1,\ell}, a_T)$, $c = \mathsf{H}(x||a||m)$, $z = (z_1^{1,\ell}, \ldots, z_k^{1,\ell}, z_T))$ of a message m under the signers' verification keys $x = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$.

Our above construction can be seen as A applying the Fiat-Shamir transform to our (k,ℓ)-PTR by delegating the computation of the underlying protocol Σ_{base} (to prove knowledge of the signing key sk_i relative to the verification key vk_i) to each signer S_i.

Informally, A is unable to forge a signature because of (1) the special soundness of the underlying Σ_{base} (i.e., each signer S_i cannot come up with an accepting transcript (a_i, c, z_i) without knowing sk_i), (2) the special soundness of the non-interactive version of our (k,ℓ)-PTR compiler of Sec. 4.1.4.4 (i.e., A cannot come up with an accepting transcript without knowing at least k accepting transcripts for Σ_{base} on the challenge $c = \mathsf{H}(x||a||m)$ and statement x, where each

[15] A will use, for each a_i containing the commitment parameters tuple $(p_0^j, p_1^j)_{j\in[\log \ell]}$, the corresponding witness $W = (w_j)_{j\in[\log \ell]}$ to generate the first-round message a_T of Π_{ord}.

of those transcripts is computed from different signing keys), (3) the ZK property of the underlying non-interactive version of Σ_{base}[16] (i.e., A cannot learn the signing key of any signer S_i by interacting with S_i). Anonymity, instead, is guaranteed by the zero-knowledge property of the non-interactive version of our (k, ℓ)-PTR.

Extension to hierarchical ring signatures. We can go beyond threshold ring signatures. Consider the following scenario as a possible example. An organization is made of ℓ_2 different sub-groups, each of them containing ℓ_1 signers. Each sub-group can approve the content of a message if k_1 members agree to sign the message. Then, organization-wide, this message is considered approved if at least k_2 of the ℓ_2 sub-groups signed such message. We name such scenario hierarchical ring signature (i.e., a "threshold-of-threshold" ring signature).

The aggregator A, in order to compute such a hierarchical ring signature, will compute a (k_2, ℓ_2)-PTR with the approach described in the previous paragraph using a (k_1, ℓ_1)-PTR as the base Σ-protocol: A will compute the first-round messages of the (k_2, ℓ_2)-PTR starting from the first-round messages of the k_2 underlying instances of the (k_1, ℓ_1)-PTR used as a base Σ-protocol. A, instead of directly communicating with the signers, will now talk to aggregators $A_1, \dots, A_{\ell_2}$, where each A_i, $i \in [\ell_2]$ is associated to a sub-group of signers $S_1^i, \dots, S_{\ell_1}^i$. At least k_2 of those A_i for $i \in \{1, \dots, \ell_2\}$ will, in turn, execute the base Σ-protocol for a (k_1, ℓ_1)-PTR by interacting with its own subset of signers $S_1^i, \dots, S_{k_1}^i$ using the technique described in the previous paragraph, with the difference that the challenge c now depends on the statement used by A for the (k_2, ℓ_2)-PTR. The very same technique can be easily extended to an arbitrary number of levels.

4.1.6 On the Security Proofs of [62]

In this section, since we defined 1-out-of-2 equivocal commitments differently w.r.t. [62], we prove the security of the $(1, \ell)$-PTRof [62] using the commitment defined in Sec. 4.1.4.2. Then, we focus on two subtle

[16]Notice that when a Σ-protocol with HVZK is compiled into a non-interactive argument system using the Fiat-Shamir transform it becomes ZK in the random oracle model.

issues in the security proofs of [62]. The first one is an inaccuracy more than an actual mistake and involves the special soundness property. The second one is more significant. In particular, the proof of the EHVZK property contained a mistake. At a certain point of the security reduction the indistinguishability of two hybrids is reduced to a property that it is actually insufficient. This is why we introduced a hiding property of the commitment scheme called trapdoornes that it is strictly stronger than the usual hiding property. In a recent update of their paper [63], Goel et al. acknowledged the issues we pointed out.

Computational special soundness of [62]. It is possible to create an extractor $\mathsf{Extract}_{1,2}$ for $\Pi_{1,2}$ using the extractor $\mathsf{Extract}'$ of the underlying protocol Π'. $\mathsf{Extract}_{1,2}$ receives in input two accepting transcripts of $\Pi_{1,2}$ $(a = (\mathtt{com}, p_0, p_1), c_0, z_0 = (z'_0, r_0, p_0, p_1))$ and $(a = (\mathtt{com}, p_0, p_1), c_1, z_1 = (z'_1, r_1, p_0, p_1))$ for the same statement x. $\mathsf{Extract}_{1,2}$ can derive the first-round messages of the underlying underlying protocol Π' by running its EHVZK simulator. Let such messages be $a_0^1 \leftarrow \mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}(x_0, c_0, z'_0)$, $a_1^1 \leftarrow \mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}(x_1, c_0, z'_0)$, $a_0^2 \leftarrow \mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}(x_0, c_1, z'_1)$, $a_1^2 \leftarrow \mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}(x_1, c_1, z'_1)$.

With non-negligible probability either $a_0^1 = a_0^2$ or $a_1^1 = a_1^2$, otherwise it is possible to break the computational fixed equivocation of CS. GGHK has computational fixed equivocation, therefore there is a negligible probability that a PPT P^* breaks the fixed equivocation property. Assume $a_0^1 \neq a_0^2$ and $a_1^1 \neq a_1^2$, the adversarial sender $\mathcal{A}$ breaks the fixed equivocation property of CS as follows:

- $\mathcal{A}$ sends $(p_0, p_1, r_0, r_1, r_0, r_1, a_0^1, a_0^2, a_1^1, a_1^2, a_0^1, a_0^2, a_1^1, a_1^2)$ to the challenger of fixed equivocation.

Since the two transcripts are accepting, it holds that $\mathsf{BindCom}(\mathsf{pp}, p_0, p_1, a_0^1, a_1^1; r_0) = \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, a_0^2, a_1^2; r_1) = \mathtt{com}$ such that $a = (\mathtt{com}, p_0, p_1)$. Thus $\mathcal{A}$ wins the fixed equivocation experiment with the same probability that $a_0^1 \neq a_0^2$ and $a_1^1 \neq a_1^2$. Since the probability that $\mathcal{A}$ wins the fixed equivocation experiment is negligible, the following holds with non-negligible probability: either $a_0^1 = a_0^2$ or $a_1^1 = a_1^2$. It follows that $\mathsf{Extract}_{1,2}$ has, with non-negligible probability, at least one pair of accepting transcripts for Π' sharing the same first-round message to give in input to $\mathsf{Extract}'$.

$\mathsf{Extract}_{1,2}$ runs $\mathsf{Extract}'$ on such transcripts and outputs whatever $\mathsf{Extract}'$ outputs. Recall that $x = (x_0, x_1)$ and $\mathsf{Extract}'$ outputs a witness w such that $(x_i, w) \in \mathcal{R}_{\mathcal{L}}$ with $i \in \{0,1\}$, which is also a valid witness for $x_0 \vee x_1$ If Π' is special sound, $\mathsf{Extract}'$ successfully extracts a witness with probability 1. Instead, if special soundness holds only computationally $\mathsf{Extract}'$, $\mathsf{Extract}$ will have a negligible probability of failure.

In [62] an hash function is used to compress the first message of Π. We consider this in the rest of the proof. Let $\mathsf{Hf} = (\mathsf{Gen}, \mathsf{H})$ be a collision resistant hash function, now the message a produced by P_0 is $a = \mathsf{H}^s((\mathtt{com}, p_0, p_1))$, where s is a random value defined in a CRS[17] and generated using algorithm Gen on the same security parameter used in Π. Let us assume that even if the adversarial prover P^* of Π cannot break the partial binding of CS and cannot break the special soundness of the underlying protocol Π', P^* can still break computational special soundness of Π. We can show that in this case there exists an adversary $\mathcal{A}$ with access to P^* that wins in $\mathsf{ExpHashColl}$ with non-negligible probability. $\mathcal{A}$ obtains the value s in the CRS and the message (a, c_0, c_1, z_0, z_1) from P^*. Looking at the transcripts (a, c_0, z_0) and (a, c_1, z_1), it cannot be that a is equal to $\mathsf{H}^s((\mathtt{com}, p_0, p_1))$ for the same $(\mathtt{com}, p_0, p_1)$, otherwise, as previously shown, it is possible to break either the special soundness of Π' or the partial binding of CS. Then there must exists two values $(\mathtt{com}, p_0, p_1)$ and $(\mathtt{com}', p_0', p_1')$ such that $\mathsf{H}^s((\mathtt{com}, p_0, p_1)) = \mathsf{H}^s(\mathtt{com}', p_0', p_1'))$. This means that $\mathcal{A}$ found a collision for Hf, thus reaching a contradiction. Since $\mathcal{A}$ can win in this experiment only with negligible probability, it follows that P^* can break computational special soundness with the same negligible probability.

EHVZK of [62]. Let $D_c^{(z)} = \Big\{(z^*, r, p_0, p_1) | r \leftarrow\!\!\$\ \mathtt{D}; z^* \leftarrow\!\!\$\ D_c^{(z^*)}; (p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta = 1)\Big\}$, where $\mathtt{D}$ is the randomness space of the un-

[17]We recall that the CRS does not need to be generated by a trusted entity since it is composed only of the description of a CRHF (e.g., SHA256 in practice).

derlying 1-out-of-2 equivocal commitment scheme[18] and pp the shared random string previously generated running $\mathsf{Setup}(1^\lambda)$. It is straightforward to see that $\Pi_{1,2}$ is EHVZK with recyclable third-round messages.

The simulator $\mathcal{S}^{\mathsf{EHVZK}}((x_0, x_1), c, (z^*, r, p_0, p_1))$ is defined as follows.

1. Compute $a_b \leftarrow \mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}(x_b, c, z^*)$, for $b \in \{0, 1\}$;
2. Compute $\mathtt{com} \leftarrow \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, a_0, a_1, c; r)$;
3. Return $(\mathtt{com}, p_0, p_1)$.

Consider the following hybrids:

$\mathcal{H}_0$: it is equivalent to the real game with the honest prover P.

$\mathcal{H}_1$: it is equivalent to $\mathcal{H}_0$ except that the protocol is not interactive and uses a challenge c randomly sampled from $\{0,1\}^\lambda$ as challenge from V. The distribution of those two hybrids is identical since c is a random value.

$\mathcal{H}_2$: it is equivalent to $\mathcal{H}_1$ except that the prover P uses c and z^* from $D_c^{(z^*)}$ as input for $\mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}$ to compute a_β in the first round. Additionally, the execution of P'_1 in the third round is removed since the value z^* used is the one taken in input from $D_c^{(z)}$. If there exists a distinguisher D_{EHVZK} that distinguishes $\mathcal{H}_1$ from $\mathcal{H}_2$ with non-negligible probability, we can define a distinguisher D'_{EHVZK} that breaks the EHVZK property of Π'. D'_{EHVZK} sends (x_β, w) to P' in the experiment $\mathsf{ExpEHVZK}_{\mathsf{P}',\mathsf{D}'_{EHVZK}}(c)$. P' returns $(a_\beta, c, \tilde{z})$.

D'_{EHVZK} computes a transcript as follows:

- $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta)$;
- $(\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta, v_\beta, p_0, p_1, \mathsf{td})$;
- $a_{1-\beta} \leftarrow \mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}(x_{1-\beta}, c, \tilde{z})$;
- $r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, a_0, a_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$.

[18]For GGHK $\mathtt{D} = \{0,1\}^{2\lambda}$.

D'_{EHVZK} sends $(a = (\texttt{com}, p_0, p_1), c, z = (\tilde{z}, r, p_0, p_1))$ to D_{EHVZK} and returns the value returned by D_{EHVZK}.

Since Π' is perfect EHVZK, $\mathcal{H}_1$ and $\mathcal{H}_2$ are identically distributed.

$\mathcal{H}_3$: it is the same as $\mathcal{H}_2$ excepts that P computes $\texttt{com} \leftarrow \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, a_0, a_1; r)$, where r is not generated by first calling $\mathsf{EquivCom}(\mathsf{pp}, \beta, a_\beta, p_0, p_1, \mathsf{td})$ and then $\mathsf{Equiv}(\mathsf{pp}, \beta, a_0, a_1, p_0, p_1, \mathsf{td}, \mathsf{aux}; \texttt{rand}_1)$ but it is taken from the input of P. If there exists a distinguisher D_{EHVZK} that distinguishes between $\mathcal{H}_2$ and $\mathcal{H}_3$ with non-negligible probability, we can define an adversary $\mathcal{A}_{trap}$ that breaks the trapdoorness property of the underlying 1-out-of-2 equivocal commitment scheme.

- Given $(x = (x_0, x_1), w)$, $\mathcal{A}_{trap}$ computes $v_b \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_b, c, z^*)$, for each $b \in \{0, 1\}$, and $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta)$ where β is from the input of P;
- $\mathcal{A}_{trap}$ sends $(a_0, a_1, p_0, p_1, \mathsf{td}, \beta)$ to $\mathsf{ExpTrap}(\lambda)$, and receives $(\texttt{com}, r)$ from $\mathsf{ExpTrap}$.
- $\mathcal{A}_{trap}$ sends $(a = (\texttt{com}, p_0, p_1), c, (z^*, r, p_0, p_1))$ to D_{EHVZK} and receives back a bit b. $\mathcal{A}_{trap}$ outputs b.

Note that $\mathcal{A}_{trap}$ perfectly simulates $\mathcal{H}_2$ when $\mathsf{ExpTrap}$ uses $\mathsf{EquivCom}$ to generate $\texttt{com}$ and then Equiv to generate r, and perfectly simulates $\mathcal{H}_3$ when $\mathsf{ExpTrap}$ uses $\mathsf{BindCom}$ together with a uniformly random r to generate $\texttt{com}$.

$\mathcal{H}_4$: it is the same as $\mathcal{H}_3$ except that the prover of $\mathcal{H}_4$ fixes the binding position to $\beta' = 1$ when calling the parameter generation algorithm Gen. If $\beta' = \beta$, the two hybrids are identically distributed. Let us consider the case in which $\beta' \neq \beta$. If there exists a distinguisher D_{EHVZK} that distinguishes between $\mathcal{H}_3$ and $\mathcal{H}_4$ with non-negligible probability, we can define an adversary $\mathcal{A}_{hid}$ that breaks the position hiding property of the 1-out-of-2 equivocal commitment scheme with non-negligible probability.

- Given $(x = (x_0, x_1), w)$, $\mathcal{A}_{hid}$ computes $v_b \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_b, c, z^*)$, for each $b \in \{0, 1\}$.

- $\mathcal{A}_{hid}$ receives (p_0, p_1, td) from ExpHid.
- $\mathcal{A}_{hid}$ sets $(a = (\mathtt{com}, p_0, p_1), c, z = (z^*, r, p_0, p_1))$, where c and z^* are the value in input to P, and $\mathtt{com}$ is generated by running $\mathsf{BindCom}$ with the messages (a_0, a_1) and the randomness r taken from the input of P. $\mathcal{A}_{hid}$ sends (a, c, z) to D_{EHVZK} and receives a bit b. $\mathcal{A}_{hid}$ outputs b.

Note that $\mathcal{A}_{hid}$ perfectly simulates $\mathcal{H}_3$ given $\beta = 0$ when ExpHid calculates (p_0, p_1, td) from $\beta = 0$, and perfectly simulates $\mathcal{H}_4$ when ExpHid calculates (p_0, p_1, td) from $\beta = 1$.

$\mathcal{H}_5$**:** it is equal to $\mathcal{H}_4$ except that the prover P takes additional inputs $(z^*, r, p_0, p_1))$ taken from $D_c^{(z)}$, and uses such inputs instead of sampling them by itself. $\mathcal{H}_4$ and $\mathcal{H}_5$ are indistinguishable since the values passed in input in $\mathcal{H}_5$ are identically distributed to the values computed in $\mathcal{H}_4$.

Note that $\mathcal{H}_5$ is identically distributed to the simulator $\mathcal{S}^{\mathsf{EHVZK}}((x_0, x_1), c, (z^*, r, p_0, p_1))$. This concludes the proof.

Achieved properties. Since the commitment scheme GGHK achieves computational fixed equivocation, perfect position hiding and perfect trapdoorness in the SRS model, the resulting protocol $\Pi_{1,2}$ of [62] is computational special sound and preserves the EHVZK flavour of Π'. The above is achieved in the SRS model.

Remark 3. We notice that in [62] they refer to the standard formulation of Σ-protocols requiring special soundness to hold for an unbounded prover. Nevertheless, they then go on by proposing an extractor that fails only with negligible probability assuming the commitment scheme is computational partial binding (i.e., a weaker version of the fixed equivocation property stating that it is infeasible to generate a commitment and accepting openings to both (m_0^1, m_1^1) and (m_0^2, m_1^2) such that $m_0^1 \neq m_0^2$ and $m_1^1 \neq m_1^2$.). It is straightforward to notice that their construction cannot achieve statistical/perfect special soundness in the classical sense since it relies on a security property that only holds computationally. We took care of this subtlety by formally defining computational special soundness in Sec. 2.3.1.1 and using such definition in the reduction presented above.

Remark 4. We also noticed an issue in the proof for EHVZK proposed in [62]. They prove EHVZK using the following hybrids:

- $\mathcal{H}^{\beta}$ that is equivalent to the real protocol, except that in the first round a_β is generated using $\mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_\beta, c, z)$;
- $\mathcal{H}^{\beta,p_0,p_1}$ that is equivalent to $\mathcal{H}^{\beta}$, except that the commitment is computed setting $\beta = 1$.

In [62], they argue that $\mathcal{H}^{\beta}$ and $\mathcal{H}^{\beta,p_0,p_1}$ are perfectly indistinguishable thanks to the perfect hiding of their commitment, but this property is not enough to prove the indistinguishability of the two hybrids. Indeed, their hiding definition did not involve the commitment opening at all, that is instead sent in the third round of the protocol. It is straightforward to see that it is possible to construct a commitment that is hiding according to the definition of [62] but reveals the biding position during the Reveal phase, thus making the two hybrids clearly distinguishable. For example, considering the Reveal Phase of the commitment scheme in Sec. 4.1.4.2, we can modify the message sent by the sender to the receiver concatenating the index of the binding position to it. It is straightforward to see that the resulting scheme is still perfect hiding according to the definition of [62]. However, this is not the only issue of the reduction presented in [62]. Indeed, the simulator just uses $\mathsf{BindCom}$ instead of $\mathsf{EquivCom}$ and Equiv, but this change is never performed in the reduction. Therefore, even if $\mathcal{H}^{\beta,p_0,p_1}$ was indistinguishable from $\mathcal{H}^{\beta}$, $\mathcal{H}^{\beta,p_0,p_1}$ does not match the distribution of the simulator as argued in [62]. For this reason we introduced a stronger hiding property called trapdoorness, which involves both the commit and the reveal phase, and requires that running $\mathsf{BindCom}$ is indistinguishable from running $\mathsf{EquivCom}$ and Equiv. Crucially, after having reduced to the trapdoorness property, we can switch the binding position by leveraging the position hiding property we defined, which only involves an adversary having access to the commitment parameters but not to the corresponding trapdoor.

4.2 Extendable Threshold Ring Signatures

Anonymity is a central requirement in several privacy-preserving technologies. Notable examples are e-voting protocols [100], anonymous authentication [91], and privacy-protecting cryptocurrencies [113]. A central cryptographic primitive that can be used to provide anonymity in applications are threshold ring signatures (cfr., Sec. 1.4.1). Optimizing threshold ring signatures in terms of efficiency, used assumptions, and size obviously produces important improvements in the corresponding applications. Moreover, extending the functionalities provided by such primitives may enable new interesting applications. In this section, we give new contributions on extendable threshold ring signatures (ETRS) with applications to anonymous count-me-in. Part of the results presented in this section will appear at PKC 2023. In the meanwhile, we published our results on ePrint [10].

4.2.1 Related Work

Threshold ring signatures were introduced by Bresson et al. [30]. They provided a construction based on RSA. The size of the signature is $\mathcal{O}(n \log n)$, where n is the size of the ring. Subsequent works proposed new constructions from a variety of assumptions focused on either relaxing the setup assumptions, reducing the signature size, or getting rid of the ROM.

Several works have signatures of size linear in n [2, 75, 95, 129], while some others proposed constructions with signature size that can be sub-linear in n [8,11,123][19], or even $\mathcal{O}(t)$ [74,90]. Some works have also focused on providing post-quantum security [2, 23, 75].

In [93], the concept of flexibility was introduced. A flexible threshold ring signature scheme allows one to modify an already created signature on a message m with threshold t and ring $\mathcal{R}$ into a new signature on message m with threshold $t + 1$ w.r.t. $\mathcal{R}$, without the intervention of the previous signers.

Usually, threshold ring signatures are formulated as an interactive protocol run among the signers. Some schemes have a weaker requirement [8, 11], where the signers just have to interact with one party

[19]In particular, [123] has size $\mathcal{O}(t\sqrt{n})$, [11] is $\mathcal{O}(t \log n)$, and [8] is $\mathcal{O}(\log n)$.

called the aggregator. After having interacted with all the signers, the aggregator just compiles all the received contributions into one threshold ring signatures which can then be publicly posted. Munch-Hansen et al. [90] presented a threshold ring signature based on RSA accumulators with size $\mathcal{O}(t)$. Their scheme also achieves flexibility. Moreover, they introduce a stronger anonymity property that demands that a signer cannot be deanonymized even by their fellow signers. In this scenario, having non-interactive signing is crucial since the deanonymization could be done by exploiting communication metadata such as the IP address. The same applies to signatures using an aggregator, unless the aggregator is trusted. Recently, Aranha et al. [5] have further enhanced the functionality of threshold ring signature by proposing extendable threshold ring signatures ETRS. ETRS are flexible and they also allow to extend the ring of a given signature without the need of any secret.

Shortcoming of ETRS Aranha et al. [5] observe how the richer flexibility of ETRS can enable more advanced forms of whistleblowing or anonymous petitions. The first signer could create a ring signature with a sufficiently large ring announcing a proposition in the signed message. After such cause becomes *public*, other parties could support the cause via extend and/or join operations. As also reported in [5], an observer who has seen signatures on an old ring $\mathcal{R}$ and on a new ring $\mathcal{R}'$ can always compute $\mathcal{R}' \setminus \mathcal{R}$, and this can help narrowing down the identity of the signers. This problem is inherent in the functionality provided by ETRS, and it worsens as t approaches the size of the ring. A clear example is the one of a signature w.r.t. ring $\mathcal{R}$ with threshold $t = n - 1$, where $n = |\mathcal{R}|$, which is transformed into a signature with threshold $t = n' - 1$ w.r.t. $\mathcal{R}', |\mathcal{R}'| = n' = n + 1$ (i.e., the threshold is increased by one and the final ring contains an additional public key of a user A). By looking at the two signatures, one can infer that one signer of the last signature either comes from $|\mathcal{R}|$ or it is A with probability $\frac{1}{2}$. In [5], the authors address this issue by proposing an anonymity definition in which the adversary is restricted to see only the signature obtained eventually, after all the extend and join operations have been applied. However, this restriction hinders the use of ETRS in real-world count-me-in applications since it bears an implicit requirement: the signers should privately interact to incrementally produce the ETRS and then only the final signature can be made public to the outside world. This means that *all* the possible advocates of a proposal should be given access to a private bulletin board where partial signatures are posted. Additionally, the abstract of [5] informally mentions the importance of *fellow signer anonymity* (FSA), a property stating that "it is often crucial for signers to remain anonymous even from their fellow signers". Such requirement was previously formally modeled in [90], but it is not captured by the anonymity definitions of [5]. Indeed, it is unclear how such property could be guaranteed when anonymity is only formulated w.r.t. an adversary who cannot see intermediate signatures (as real signers would) and does not have the secret key of any of the signers (as in the definition of [5]).

4.2.2 Our Contributions

In this chapter, we address the aforementioned shortcomings of ETRS. First, we propose a stronger security definition that guarantees anonymity even against adversaries that see the full "evolution" of a signature. Second, we propose a new ETRS construction that achieves our strong anonymity definition, and also improves in efficiency over previous work (cfr., Table 4.2 and 4.3). Our construction relies on extendable non-interactive witness indistinguishable proof of knowledge (ENIWI PoK), a novel technical tool that we formalize and construct, and that may be of independent interest. In what follows, we present our contributions in more detail.

Stronger anonymity for ETRS. Even though certain leaks are inherent when the adversary gets to see several ETRS, one should aim at building a scheme which leaks nothing more than that. To this regard, we start from the anonymity definition proposed in [5] and we make it stronger as follows. We allow the adversary $\mathcal{A}$ to see all the ETRS that led to the final signature. In a nutshell, $\mathcal{A}$ outputs two sequences of operations which at every step lead to an ETRS on the same message, with the same ring, and the same threshold in both sequences. The challenger $\mathcal{C}$ picks one of such sequences at random, executes it, and gives to $\mathcal{A}$ the corresponding outputs of each step. We then require that $\mathcal{A}$ only has a negligible advantage in guessing which sequence was applied. We also propose a security game that models fellow signer anonymity for ETRS.

Constructing ETRS. In [5], two constructions of ETRS are proposed: the first one is obtained from extendable same-message linkable ring signatures (SMLERS)[20], while the second one is constructed from signatures of knowledge (SoK) for the discrete log relation, public key encryption (PKE), and the discrete log assumption. The first scheme achieves our stronger anonymity notion but suffers quite high complexity; for instance, the signature size is $\mathcal{O}(tn)$. The second scheme in [5] is more compact but does not fulfill our stronger anonymity notion. Indeed, anyone who sees an ETRS before and after a join operation can easily pinpoint the exact signer who joined the signature (see Sec. 4.2.3 for more details). It follows that such scheme is also not fellow signer anonymous, since no secret key is required to carry out the above attack.

We construct an ETRS which fulfills our stronger anonymity definition and is also fellow signer anonymous. As shown in Tab. 4.2 and Tab. 4.3, our ETRS also generally improves the constructions given in [5] in terms of both time complexity and signature size. In Sec. 4.2.3, we give a high-level overview of both ETRS presented in [5]. To build our ETRS, we introduce the notion of ENIWI PoK, which may be of independent interest. We then show how to build an ETRS from an ENIWI PoK for a hard relation, and an IND-CPA homomorphic public key encryption scheme.

ENIWI PoKs. In [38], Chase et al. examined notions of malleability for non-interactive proof systems. They defined the notion of allowable transformation $T = (T_x, T_w)$ w.r.t. a relation $\mathcal{R}$. A transformation is allowable w.r.t. $\mathcal{R}$ if on input $(x, w) \in \mathcal{R}$ it gives as output

[20]SMLERS were introduced in [5] as well. A SMLERS is a ring signature which is also extendable. In addition, it allows to link two signatures produced by the same signer on the same message, even on different rings. The SMLERS of [5] is obtained from signatures of knowledge for the discrete log relation, collision-resistant hash functions, and the discrete log assumption.

Scheme	**Size**	**Anonymity**	**FSA**
SMLERS [5]	$\mathcal{O}(tn)$	Strong	Yes
DL + SoK + PKE [5]	$\mathcal{O}(N)$	Weak	No
Ours	$\mathcal{O}(n)$	Strong	Yes

Table 4.2: Comparison of signature size and anonymity guarantees of our ETRS and the ones presented in [5]. We use n to indicate the size of the ring and t to indicate the threshold. In the DL + SoK + PKE construction of [5] the signature size depends on a fixed upper bound on the ring size N. We say that a scheme achieves weak anonymity if it achieves the anonymity property of [5], while we say that a scheme achieves strong anonymity if our stronger anonymity definition is satisfied. FSA stands for fellow signer anonymity.

Scheme	**Sign**	**Join**	**Extend**	**Verify**
SMLERS [5]	$\mathcal{O}(tn)$	$\mathcal{O}(n)$	$\mathcal{O}(tn)$	$\mathcal{O}(tn)$
DL + SoK + PKE [5]	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$
Ours	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Table 4.3: Comparison of time complexities of our ETRS and the ones presented in [5].

$(T_x(x) = x', T_w(w) = w') \in \mathcal{R}$. Then, a proof system is said to be malleable w.r.t. an allowable transformation $T = (T_x, T_w)$, if there exists a poly-time algorithm that on input the initial statement x, the transformation T, and an accepting proof Π, gives an accepting proof Π' for the transformed statement x'. They also considered more complex transformations including n statements and proofs. They showed that Groth-Sahai (GS) proofs [72] are malleable w.r.t. the language of sets of pairing product equations and they define a set of elementary allowable transformations which can be used to build more complex ones, including conjunctions and disjunctions. They also observed that since GS is re-randomizable, a transformation of a proof followed by its re-randomization is indistinguishable from a proof computed from scratch for statement x' using witness w'. They called this property derivation privacy.

In this paper, we further explore the notion of malleability for non-interactive witness indistinguishable (NIWI) proofs of knowledge (PoKs) in the context of threshold relations. A threshold relation $\mathcal{R}_t$

is defined w.r.t. a relation $\mathcal{R}$ as $\mathcal{R}_t = \{(x = (k, x_1, \ldots, x_n), w = ((w_1, \alpha_1), \ldots, (w_k, \alpha_k)))|1 \leq \alpha_1 < \ldots < \alpha_k \leq n \wedge \ \forall\ j \in [k] : (x_{\alpha_j}, w_j) \in \mathcal{R}\}$. Let $\mathcal{L}_t$ be the corresponding NP language. In words, the prover wants to prove it has k witnesses for k *different* statements out of n statements. The transformations we explore are extend and add operations:

- **Extend**: transform a proof for $(k, x_1, \ldots, x_n) \in \mathcal{L}_t$ into a proof for $(k, x_1, \ldots, x_n, x_{n+1}) \in \mathcal{L}_t$.
- **Add**: transform a proof for $(k, x_1, \ldots, x_n) \in \mathcal{L}_t$ into a proof for $(k + 1, x_1, \ldots, x_n) \in \mathcal{L}_t$.

While the extend operation can be realized without using any private input of the "previous" prover, as modelled in [38], the same does not hold for the add operation. Indeed, thanks to extractability, an accepting proof for $(k + 1, x_1, \ldots, x_n) \in \mathcal{L}_t$ can only be generated by the prover, except with negligible probability, using $k + 1$ witnesses for $k + 1$ different statements out of all the n statements. It follows that the add transformation must require a witness for statement x_i, with index $i \in [n]$ that was not previously used, and it cannot produce an accepting proof for the updated statement on input a witness for a previously used index. It is straightforward to notice that this fact could be used to check whether or not a given witness was used in the proof, thus violating witness indistinguishability.

Therefore, we put forth the new notion of ENIWI PoK. In an ENIWI PoK, when the prover computes a proof Π for a statement $x = (k, x_1, \ldots, x_n)$, it also outputs a list of auxiliary values $\mathsf{AUX} = (\mathsf{aux}_1, \ldots, \mathsf{aux}_n)$. The auxiliary value aux_i will be later used to perform the add operation via an additional algorithm called PrAdd. PrAdd, on input an accepting proof Π for $(k, x_1, \ldots, x_n) \in \mathcal{L}_t$, a witness w_i for a not previously used index i s.t. $(x_i, w_i) \in \mathcal{R}_{\mathcal{L}}$, and the corresponding auxiliary value aux_i, outputs a proof Π' for $(k + 1, x_1, \ldots, x_n) \in \mathcal{L}_t$. Analogously, there is an additional algorithm $\mathsf{PrExtend}$ that is used to perform the extend operation. $\mathsf{PrExtend}$ does not require any auxiliary value. $\mathsf{PrExtend}$, on input an accepting proof for $(k, x_1, \ldots, x_n) \in \mathcal{L}_t$, and a statement x_{n+1}, outputs a proof Π' for $(k, x_1, \ldots, x_{n+1}) \in \mathcal{L}_t$ and the auxiliary value aux_{n+1} related to statement x_{n+1}. The auxiliary value aux_{n+1} can later be used to perform an add operation using witness w_{n+1} s.t. $(x_{n+1}, w_{n+1}) \in \mathcal{R}_{\mathcal{L}}$. The verification algorithm is left unaltered and does not take any auxiliary value in input.

Similarly to derivation privacy, we require that the outputs of both the extend and add operations followed by a re-randomization are indistinguishable from proofs created using the regular prover algorithm. Regarding witness indistinguishability, we have to treat the auxiliary values in a special manner. Indeed, giving out all the auxiliary values would at least reveal the indices of the used witnesses. Therefore, we propose a new notion called extended witness indistinguishability. In this notion, the adversary $\mathcal{A}$ samples a $x = (k, x_1, \ldots, x_n)$ and two witnesses w^i as $((w_1^i, \alpha_1^i) \ldots, (w_k^i, \alpha_k^i))$, s.t. $(x, w^i) \in \mathcal{R}_t$ for $i \in \{0, 1\}$. Then, the challenger $\mathcal{C}$ outputs a proof computed using one of the two witnesses, but it only gives to $\mathcal{A}$ a *subset* of all the auxiliary values. Such subset includes the auxiliary values only related to certain indices, namely $(\{1, \ldots, n\} \setminus (\{\alpha_1^0, \ldots, \alpha_n^0\} \cup \{\alpha_1^1, \ldots, \alpha_n^1\})) \cup (\{\alpha_1^0, \ldots, \alpha_n^0\} \cap \{\alpha_1^1, \ldots, \alpha_n^1\})$. In words, those are the auxiliary values related to the indices for which one of the following conditions holds: (i) the index was not used in either w^0 or w^1; (ii) the index was used in both w^0 and w^1. We require that $\mathcal{A}$ has negligible advantage in guessing whether w^0 or w^1 was used to create the proof. The idea is that if we build upon a NIWI and if the auxiliary values are only tied to the *indices* of the used witness and not to their concrete values, then giving the auxiliary values for the "irrelevant" positions to $\mathcal{A}$ does not give $\mathcal{A}$ any advantage. Although it could seem a cumbersome notion, ENIWI is enough to obtain strongly anonymous ETRS, and could possibly have other applications.

High-level overview of our ENIWI. We propose an ENIWI for the base relation $\mathcal{R}$ of pairing product equations (PPEs) in which all the variables are elements of group two, public constants are either paired with secret values or with the public generator, and the target element is the neutral element.

We build our ENIWI from GS proofs. GS is a commit-and-prove system where secret variables are first committed and the prover algorithm takes as input the committed values as well as the commitments randomnesses to create some proof elements. The proof can be verified on input the statement, the commitments, and proof elements.

We first modify known techniques to get disjunctions of PPEs [31, 69] into a technique to get proofs of partial satisfiability of k out of n PPEs. Such transformation modifies the starting PPEs via some additional variables $\hat{M}_i$ with $i \in [n]$ s.t. k of the PPEs are left unaltered while $n-k$ of them now admit the trivial solution, thus allowing for simulation. The value of $\hat{M}_i$ is constrained to two values, depending on whether or not the proof for the i-th equation should be simulated. We then observe that such proofs can be turned into an ENIWI provided with the extend and the add operations. The auxiliary values can be seen as the commitment openings related to such variables which allow to replace an $\hat{M}_i$ allowing for simulation with a new one preventing simulation. The idea is that to perform the add operation, the old commitment to a variable $\hat{M}_i$ would be replaced with a fresh one. Then, aux_i would allow to erase from the proof element the contribution related to the old committed variable and to subsequently put in the contribution of the freshly committed variable. The extend operation is more straightforward since it does not need to erase any contribution, but only to add the contribution of a new variable. At a high level, extended witness indistinguishability is achieved since the $\hat{M}_i$ variables are only tied to the particular equation being simulated or not, but not to the actual value of any of the variables. Proofs can also be re-randomized leveraging the re-randomizability of GS and by appropriately updating the auxiliary values after the re-randomization.

High-level overview of our ETRS. To get an ETRS, we just need a way to turn an ENIWI in a signature scheme preserving its extendability properties. In [57], it is shown how to create a signature of knowledge (SoK) from a NIWI PoK in the random oracle model (ROM). In a nutshell, the message is hashed to produce the CRS which is then used to prove the statement of the SoK. The resulting proof constitutes the signature. We leverage their technique to create an ETRS starting from an ENIWI PoK. The idea is that since the transformation given in [57] just modifies how the CRS is generated, we are able to replace the NIWI PoK with an ENIWI PoK to get an ETRS instead of a regular signature. In our ETRS, the i-th signer has as public key a statement x_i for a hard relation $\mathcal{R}_{\mathcal{L}}$ for which it exists an ENIWI, along with the public key $\mathsf{pk}_{\mathsf{e}}^i$ of an IND-CPA public key encryption scheme (PKE)

which is homomorphic w.r.t. the update operation of the auxiliary values. The corresponding secret key is w_i s.t. $(x_i, w_i) \in \mathcal{R}_{\mathcal{L}}$, along with the secret key of the encryption scheme sk_e^i. The first signer S hashes the message m to get the CRS, then S uses her own witness to create a proof for $(1, x_1, \ldots, x_n) \in \mathcal{R}_t$. By creating such proof, the signer will also get auxiliary values $(\mathsf{aux}_1, \ldots, \mathsf{aux}_n)$. Since publishing the auxiliary values in the clear would reveal the identity of the signer, each individual aux_i is encrypted using the public key of the i-th signer. A new signer willing to join will decrypt aux_i and run PrAdd to update the proof. To extend the ring, it suffices to run $\mathsf{PrExtend}$ to update the proof. Finally, to ensure anonymity we exploit the fact that ENIWI PoKs are re-randomizable. We re-randomize all the proofs after running either PrAdd or $\mathsf{PrExtend}$. We additionally exploit the homomorphic property of the encryption scheme to update the auxiliary values after each re-randomization. We prove the security of our ETRS in the ROM.

Both the constructions presented in [5] use SoKs for the discrete log relation as a building block without specifying a concrete instantiation. Whether they require the ROM or not depends on whether there exists a practical[21] SoK without random oracles for that relation. The authors also provide an implementation in which they use the Schnorr identification scheme with the Fiat-Shamir transform as a SoK. Such SoK requires the ROM. In our ETRS, all operations require linear time in n as the number of equations to be proved linearly depends on n. Additionally, GS proofs have constant size for each type of equation, therefore the size of the ETRS is $\mathcal{O}(n)$. Note that both time complexity and signature size do not depend on t.

4.2.3 A Closer Look to the Results of [5]

In this section, we give a high-level overview of the two ETRS presented in [5].

[21] Chase and Lysyanskaya [39] proposed a generic construction under standard complexity assumptions in the common random string model, but it is not practical since it uses general non-interactive zero-knowledge (NIZK) proofs.

SMLERS-based ETRS. The first ETRS given in [5] is based on same-message linkable ring signatures (SMLERS). In a nutshell, a SMLERS is a ring signature which is provided with the Extend algorithm. In addition, it allows to link two signatures produced by the same signer on the same message, even on different rings. This is done via a linkability tag that is uniquely determined by the signer and the message. To construct an ETRS, it simply suffices to concatenate several SMLERS on the same message, provided they do not carry the same linkability tag, and to extend all the SMLERS to have the same ring. Their SMLERS has size $\mathcal{O}(n)$, where n is the size of the ring. Therefore, the compiled ETRS has size $\mathcal{O}(tn)$. The same reasoning applies to the time complexity of most of the operations. Although the anonymity property was proven in [5] according to their weaker definition, it seems reasonable to assume that this construction could be proven secure under our stronger definition without much effort.

DL+SoK+PKE ETRS. The second construction of ETRS proposed in [5] works with a prime order group, with two public group elements (g, H), and a signature of knowledge for relation $\mathcal{R}$ for knowledge of the discrete logarithm either of a certain value h or of a pk. When the first signature is generated, the signer generates N points of the form $(x_i, td_i) \in \mathbb{Z}_p^2$. All these points define a unique polynomial p of degree N such that $p(0) = dlog(H)$ and $p(x_i) = td_i$ for all $i \in [N]$. The discrete log of H is not known, but this polynomial can be interpolated in the exponent to obtain a polynomial f s.t. $f(x) = g^{p(x)}$. To either sign or join a signature, the signer has to produce a signature of knowledge for $\mathcal{R}$ using a random point $(x, y = g^{p(x)})$ with $x \notin \{x_i\}_{i\in[N]}$. If the DL problem is hard, the signer cannot know the discrete log of y and thus the signature of knowledge must satisfy the second clause of $\mathcal{R}$ which requires proving knowledge of the secret key. To extend a signature instead, an $x \in \{x_i\}_{i\in[N]}$ will be used, so that the corresponding trapdoor can be used to satisfy the first clause of the relation. The used pair (x_i, td_i) is removed from the list of trapdoors. However, to make the owner of the public key pk able to join at a later time, along with the signature of knowledge, the point x_i is posted together with an encryption of td_i under pk. As a result, the owner of pk can decrypt td_i and put it back to the list of trapdoors, before producing a fresh signature of knowledge using her secret key as a witness. Although neither space nor time complexities depend on the threshold t, they all take time and space proportional to N even though the actual ring size $n << N$. In particular, every operation requires to interpolate the polynomial in the exponent, which takes quadratic time in the number of interpolated points N. Regarding anonymity, it is pretty straightforward to observe that Def. 26 is not satisfied. Indeed, after a join operation, a "trapdoored" point x_i which was related to a signature of knowledge for a certain key pk_i goes back in the trapdoor list, and it is replaced by a new random "non-trapdoored" point. This movement of x_i from one list to another clearly indicates that pk_i has joined the signature.

4.2.4 Preliminaries

We work over bilinear groups $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathsf{Gen}(1^\lambda)$. $\mathsf{Gen}(1^\lambda)$ is a generator algorithm that on input the security parameter, outputs the description of a bilinear group. We call such description group key gk. $\hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}$ are prime p order groups, $\hat{g}, \check{h}$ are generators of $\hat{\mathbb{G}}, \check{\mathbb{H}}$ respectively, and $e : \hat{\mathbb{G}} \times \check{\mathbb{H}} \rightarrow \mathbb{T}$ is a non-degenerate bilinear map. We will use additive notation for the group operations and multiplicative notation for the bilinear map e.

Assumption 4 (DDH)**.** *The Decisional Diffie-Hellman (DDH) holds in $\hat{\mathbb{G}}$ if for all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ distinguishes the two distributions $(\hat{g}, \xi\hat{g}, \rho\hat{g}, \xi\rho\hat{g})$ and $(\hat{g}, \xi\hat{g}, \rho\hat{g}, \kappa\hat{g})$, where $\xi, \rho, \kappa \leftarrow \mathbb{Z}_p$ is negligible. Tuples of the form $(\hat{g}, \xi\hat{g}, \rho\hat{g}, \xi\rho\hat{g})$ are called Diffie-Hellman (DH) tuples. The DDH problem in $\check{\mathbb{H}}$ is defined in a similar way.*

Assumption 5 (SXDH)**.** *The Symmetric eXternal Diffie-Hellman (SXDH) assumption holds relative to* Gen *if there is no PPT adversary $\mathcal{A}$ that breaks the DDH problem in both $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ for* $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathsf{Gen}(1^\lambda)$.

Assumption 6 (Double Pairing Fixed Term Assumption)**.** *We say the double pairing fixed term assumption holds relative to $\hat{\mathbb{G}}$ if for*

$\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathsf{Gen}(1^\lambda)$, *and for all PPT adversaries* $\mathcal{A}$ *we have*

$$\Pr\left[\hat{a}, \hat{b} \leftarrow\!\!\$\ \hat{\mathbb{G}} \setminus (\hat{0}, \hat{0}); \check{b}' \leftarrow \mathcal{A}(\mathsf{gk}, \hat{a}, \hat{b}) : \check{b}' \in \check{\mathbb{H}}, \hat{a} \cdot \check{h} + \hat{b} \cdot \check{b}' = 0_{\mathbb{T}}\right] \leq \mathsf{negl}(\lambda).$$

Lemma 5. *If the double pairing fixed term assumption holds for* gk*, then the Decisional Diffie-Hellman assumption holds for* $\hat{\mathbb{G}}$.

Proof. We build an adversary $\mathcal{B}$ for the DDH assumption which makes black-box use of $\mathcal{A}$. $\mathcal{B}$ gets the DDH challenge $(\hat{a}, \hat{b}, \hat{c})$. If $\hat{a} = \hat{0}$ or $\hat{b} = \hat{0}$ or $\hat{c} = \hat{0}$, $\mathcal{B}$ aborts. This event occurs with negligible probability. $\mathcal{B}$ sets the challenge for $\mathcal{A}$ to be $(\hat{c}, \hat{b})$. With probability ϵ, $\mathcal{B}$ gets back $\check{b}' \in \check{\mathbb{H}}$ s.t. $\hat{c} \cdot \check{h} + \hat{b} \cdot \check{b}' = 0_{\mathbb{T}}$ from $\mathcal{A}$. $\mathcal{B}$ outputs 1 iff $\hat{a} \cdot \check{h} + \hat{g} \cdot \check{b}' = 0_{\mathbb{T}}$, and 0 otherwise. If $(\hat{a}, \hat{b}, \hat{c})$ is a DH tuple, $\mathcal{B}$ outputs 1 with probability exactly ϵ. In the other case, since $\hat{a} \neq \hat{0}$, $\hat{b} \neq \hat{0}$, and $\hat{c} \neq \hat{0}$, there is no trivial $\check{b}'$ s.t. $\hat{a} \cdot \check{h} + \hat{g} \cdot \check{b}' = 0_{\mathbb{T}}$. □

4.2.4.1 ElGamal Encryption

The ElGamal encryption scheme is a public key encryption scheme with the following algorithms. The public parameters pp produced by Setup are implicitly available to all other algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: on input the security parameter, sample a cyclic group $\hat{\mathbb{G}}$ of prime order p, a generator $\hat{g}$. Output $\mathsf{pp} = (\hat{\mathbb{G}}, \hat{g})$.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$: sample an element $\zeta \leftarrow\!\!\$\ \mathbb{Z}_p^*$. Define public key as $\mathsf{pk} = \hat{\boldsymbol{v}} = (\zeta\hat{g}, \hat{g})^\top \in \hat{\mathbb{G}}^{2\times1}$ and $\mathsf{sk} = \boldsymbol{\zeta} = (-\zeta^{-1}, 1)$. Output $(\mathsf{pk}, \mathsf{sk})$.

- $\hat{\mathbf{a}} \leftarrow \mathsf{Enc}(\hat{m}, \mathsf{pk})$: with input the public key and a message $\hat{m} \in \hat{\mathbb{G}}$, sample $r \leftarrow\!\!\$\ \mathbb{Z}_p$ and output ciphertext $\hat{\mathbf{a}} = \boldsymbol{e}^\top\hat{m} + \hat{\boldsymbol{v}}r \in \hat{\mathbb{G}}^{2\times1}$, where $\boldsymbol{e} = (0, 1)$.

- $\hat{m} \leftarrow \mathsf{Dec}(\hat{\mathbf{a}}, \mathsf{sk})$: with input the secret key and a ciphertext $\mathbf{a} \in \hat{\mathbb{G}}^{2\times1}$, output $\hat{m} = \boldsymbol{\zeta}\hat{\mathbf{a}}$.

The ElGamal encryption scheme is also homomorphic, with the function f being the group operation. In more detail:

- $\mathsf{a}' \leftarrow \mathsf{Eval}(\mathsf{a}_1, \hat{m}_2, \mathsf{pk})$: compute $\mathsf{a}_\mathbf{2} = \mathsf{Enc}(\hat{m}_2, \mathsf{pk})$, output $\mathsf{a}' = \mathsf{a}_1 + \mathsf{a}_2$. If the ciphertexts contained messages $\hat{m}_1$ and $\hat{m}_2$, the output ciphertext will contain message $\hat{m}_1 + \hat{m}_2$.

The ElGamal encryption is IND-CPA secure if the DDH assumption holds in $\hat{\mathbb{G}}$. Additionally, ciphertexts updated with Eval are identically distributed to freshly generated ciphertexts.

4.2.5 Groth-Sahai Proofs

The Groth-Sahai proof system [72] is a proof system for the language of satisfiable equations (of types listed below) over a bilinear group $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathsf{Gen}(1^\lambda)$. The prover wants to show that there is an assignment of all the variables that satisfies the equation. Such equations can be of four types:

Pairing-product equations (PPE): For public constants $\hat{a}_j \in \hat{\mathbb{G}}$, $\check{b}_i \in \check{\mathbb{H}}$, $\gamma_{ij} \in \mathbb{Z}_p$, $t_\mathbb{T} \in \mathbb{T}$:

$$\sum_i \hat{x}_i \cdot \check{b}_i + \sum_j \hat{a}_j \cdot \check{y}_j + \sum_i \sum_j \gamma_{ij} \hat{x}_i \cdot \check{y}_j = t_\mathbb{T}.$$

Multi-scalar multiplication equation in $\hat{\mathbb{G}}$ (ME$_{\hat{\mathbb{G}}}$): For public constants $\hat{a}_j \in \hat{\mathbb{G}}$, $b_i \in \mathbb{Z}_p$, $\gamma_{ij} \in \mathbb{Z}_p$, $\hat{t} \in \hat{\mathbb{G}}$:

$$\sum_i \hat{x}_i b_i + \sum_j \hat{a}_j y_j + \sum_i \sum_j \gamma_{ij} \hat{x}_i y_j = \hat{t}.$$

Multi-scalar multiplication equation in $\check{\mathbb{H}}$ (ME$_{\check{\mathbb{H}}}$): For public constants $a_j \in \mathbb{Z}_p$, $\check{b}_i \in \check{\mathbb{H}}$, $\gamma_{ij} \in \mathbb{Z}_p$, $\check{t} \in \check{\mathbb{H}}$:

$$\sum_i x_i \check{b}_i + \sum_j a_j \check{y}_j + \sum_i \sum_j \gamma_{ij} x_i \check{y}_j = \check{t}.$$

Quadratic equation in $\mathbb{Z}_p$ (QE): For public constants $a_j \in \mathbb{Z}_p$, $b_i \in \mathbb{Z}_p$, $\gamma_{ij} \in \mathbb{Z}_p$, $t \in \mathbb{Z}_p$:

$$\sum_i x_i b_i + \sum_j a_j y_j + \sum_i \sum_j \gamma_{ij} x_i y_j = t.$$

Here, we formalize the GS proof system as in [54]. The GS proof system is a commit-and-prove system. Each committed variable is also provided with a public label that specifies the type of input (i.e., scalar or group element). Accordingly, the prover algorithm takes as input a label L which indicates the type of equation to be proved (i.e., $L \in \{\mathrm{PPE}, \mathrm{ME}_{\hat{\mathbb{G}}}, \mathrm{ME}_{\check{\mathbb{H}}}, \mathrm{QE}\}$). GS features the following PPT algorithms, the common reference string crs and the group key gk are considered as implicit input of all the algorithms.

- $\mathsf{crs} \leftarrow\$ \mathsf{CRSSetup}(\mathsf{gk})$: on input the group key, output the common reference string.
- $(l, c) \leftarrow \mathtt{com}(l, w; r)$: return a commitment (l, c) to message w according to the label l and randomness r.
- $\pi \leftarrow \mathsf{Prove}(L, x, (l_1, w_1, r_1), \ldots, (l_n, w_n, r_n))$: consider statement x as an equation of type specified by L, and on input a list of commitment openings produce a proof π.
- $0/1 \leftarrow \mathsf{PrVerify}(x, (l_1, c_1), \ldots, (l_n, c_n), \pi)$: given committed variables, statement x, and proof π, output 1 to accept and 0 to reject.
- $((l_1, c'_1), \ldots, (l_n, c'_n), \pi') \leftarrow \mathsf{RandPr}(L, (l_1, c_1), \ldots, (l_n, c_n), \pi; r)$: on input equation type specified by L, a list of commitments, a proof π, and a randomness r, output a re-randomized proof along with the corresponding list of re-randomized commitments.

GS can be also used to prove that a set of equations S, with possibly shared variables across the equations, has a satisfying assignment. To do so, the prover has to reuse the same commitments for the shared variables while executing the Prove algorithm for each individual equation. The above description can also fit the interface of NIWI PoK (cfr., Sec. 2.3.1.2). Indeed, the Prove algorithm can just launch the $\mathtt{com}$ and the Prove algorithm above with the appropriate labels, and return as a proof both the commitments and the proof elements. Similarly, the $\mathsf{PrVerify}$ and RandPr algorithms of the NIWI PoK interface have just to appropriately parse their inputs and call the $\mathsf{PrVerify}$ and RandPr algorithms described above.

The GS proof system is proved to be a NIWI for all types of the above equations under the SXDH assumption. In addition, it is a NIWI PoK for all equations involving solely group elements. To be more specific, Escala and Groth formulated the notion of F-knowledge [54] (i.e., a variation of adaptive extractable soundness, see Def. 17 in Sec. 2.3.1.2) for a commit-and-prove system. In a nutshell, it requires the existence of an Ext_2 algorithm that, on input a valid commitment and the extraction key produced by Ext_1, outputs a function F of the committed value. They prove that GS enjoys F-knowledge. For commitments to group elements, F is identity function. Regarding commitments to scalars, F is a one-way function that uniquely determines the committed value.

4.2.5.1 Internals of GS Proofs

In [54], the authors provide a very fine-grained description of GS proofs. In this description, we report only the aspects that are relevant to our constructions. It is possible to write the equations of Sec. 4.2.5 in a more compact way. Consider $\hat{\boldsymbol{x}} = (\hat{x}_1, \ldots, \hat{x}_m)$ and $\check{\boldsymbol{y}} = (\check{y}_1, \ldots, \check{y}_n)$, which may be both public constants (i.e., written before as $\hat{a}_j, \check{b}_i$) or secret values. Let $\Gamma = \{\gamma_{ij}\}_{i=1,j=1}^{m,n} \in \mathbb{Z}_p^{m \times n}$. We can now write a PPE as $\hat{\boldsymbol{x}}\Gamma\check{\boldsymbol{y}} = t_{\mathbb{T}}$. Similarly, a $\mathrm{ME}_{\hat{\mathbb{G}}}$, a $\mathrm{ME}_{\check{\mathbb{H}}}$, and a QE can be written as $\hat{\boldsymbol{x}}\Gamma\boldsymbol{y} = \hat{t}$, $\boldsymbol{x}\Gamma\check{\boldsymbol{y}} = \check{t}$, and $\boldsymbol{x}\Gamma\boldsymbol{y} = t$. This holds for $\hat{\boldsymbol{x}} \in \hat{\mathbb{G}}^{1\times m}, \check{\boldsymbol{y}} \in \check{\mathbb{H}}^{n\times 1}, \boldsymbol{x} \in \mathbb{Z}_p^{1\times m}, \boldsymbol{y} \in \mathbb{Z}_p^{n\times 1}$. Additionally, for equations of type $\mathrm{ME}_{\hat{\mathbb{G}}}$, $\mathrm{ME}_{\check{\mathbb{H}}}$, and QE, we can, without loss of generality, assume the target element to be the neutral element. For PPE we will restrict ourselves to the case in which $t_{\mathbb{T}} = 0_{\mathbb{T}}$, and no public constants are paired with each other, unless one of the two is a generator specified in the public parameters. The structure of the crs is clear from Fig. 4.3, where the Ext_1 algorithm is shown.

In Fig. 4.4, we report the commitment labels and corresponding commit algorithm that are of interest for this work. In Fig. 4.5, a list of the possible commitment labels for each equation type.

In Fig. 4.6 and in Fig. 4.7, we report the prover and verifier algorithm respectively. Finally, in Fig. 4.8 we report a description of the proof re-randomization algorithm.

```
$(\mathsf{crs}, xk) \leftarrow \mathsf{Ext}_1(\mathsf{gk})$
Parse $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$
$\rho \leftarrow\!\$\ \mathbb{Z}_p, \xi \leftarrow\!\$\ \mathbb{Z}_p^*$ and $\sigma \leftarrow\!\$\ \mathbb{Z}_p, \psi \leftarrow\!\$\ \mathbb{Z}_p^*$
$\hat{\boldsymbol{v}} = (\xi\hat{g}, \hat{g})^\top$ and $\check{\boldsymbol{v}} = (\psi\check{h}, \check{h})$
$\hat{\boldsymbol{w}} = \rho\hat{\boldsymbol{v}}$ and $\check{\boldsymbol{w}} = \sigma\check{\boldsymbol{v}}$
$\hat{\boldsymbol{u}} = \hat{\boldsymbol{w}} + (\hat{0}, \hat{g})^\top$ and $\check{\boldsymbol{u}} = \check{\boldsymbol{w}} + (\check{0}, \hat{g})$
$\boldsymbol{\xi} = (-\xi^{-1} \mod p, 1)$ and
$\boldsymbol{\psi} = (-\psi^{-1} \mod p, 1)^\top$
$\mathsf{crs} = (\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \check{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}})$
$xk = (\boldsymbol{\xi}, \boldsymbol{\psi})$
return $(\mathsf{crs}, xk)$
```

Figure 4.3: Generation of the CRS along with the extraction key in the GS proof system.

4.2.6 Extendable Threshold Ring Signature

A non-interactive extendable threshold ring signature scheme ETRS is defined as a tuple of six PPT algorithms $\mathsf{ETRS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Join}, \mathsf{Extend})$, where the public parameters pp produced by Setup are implicitly available to all the other algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: on input the security parameter, outputs public parameters pp.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$: generates a new public and secret key pair.
- $\sigma \leftarrow \mathsf{Sign}(m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{sk})$: returns a signature with threshold $t = 1$ using the secret key sk corresponding to a public key pk_i with $i \in \mathcal{R}$.
- $0/1 \leftarrow \mathsf{Verify}(t, m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \sigma)$: verifies a signature σ for the message m against the public keys $\{\mathsf{pk}_i\}_{i\in\mathcal{R}}$ with threshold t. Outputs 1 to accept, and 0 to reject.
- $\sigma' \leftarrow \mathsf{Join}(m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{sk}, \sigma)$: it takes as input a signature σ for message m produced w.r.t. ring $\mathcal{R}$ with threshold t, and the new

Input	Randomness	Output
$\mathsf{pub}_{\hat{\mathbb{G}}}, \hat{x}$	$r = 0, s = 0$	$\hat{\boldsymbol{c}} = \boldsymbol{e}^\top \hat{x}$
$\mathsf{com}_{\hat{\mathbb{G}}}, \hat{x}$	$r, s \leftarrow\!\!\$\ \mathbb{Z}_p$	$\hat{\boldsymbol{c}} = \boldsymbol{e}^\top \hat{x} + \hat{\boldsymbol{v}} r + \hat{\boldsymbol{w}} s$
$\mathsf{base}_{\hat{\mathbb{G}}}, \hat{g}$	$r = 0, s = 0$	$\hat{\boldsymbol{c}} = \boldsymbol{e}^\top \hat{g}$
$\mathsf{sca}_{\hat{\mathbb{G}}}, x$	$r \leftarrow\!\!\$\ \mathbb{Z}_p, s = 0$	$\hat{\boldsymbol{c}} = \hat{\boldsymbol{u}} x + \hat{\boldsymbol{v}} r$

Input	Randomness	Output
$\mathsf{pub}_{\check{\mathbb{H}}}.\check{y}$	$r = 0, s = 0$	$\check{\boldsymbol{d}} = \check{y} \boldsymbol{e}$
$\mathsf{com}_{\check{\mathbb{H}}}, \check{x}$	$r, s \leftarrow\!\!\$\ \mathbb{Z}_p$	$\check{\boldsymbol{d}} = \check{y} \boldsymbol{e} + r \check{\boldsymbol{v}} + s \check{\boldsymbol{w}}$
$\mathsf{base}_{\check{\mathbb{H}}}, \check{h}$	$r = 0, s = 0$	$\check{\boldsymbol{d}} = \check{h} \boldsymbol{e}$
$\mathsf{sca}_{\check{\mathbb{H}}}, y$	$r \leftarrow\!\!\$\ \mathbb{Z}_p, s = 0$	$\check{\boldsymbol{d}} = y \check{\boldsymbol{u}} + r \check{\boldsymbol{v}}$

Figure 4.4: GS commit labels and corresponding commit algorithm, $\boldsymbol{e} = (0, 1)$.

L	Labels for x_i, $i \in [m]$	Labels for y_j, $j \in [n]$
PPE	$\mathsf{base}_{\hat{\mathbb{G}}}, \mathsf{pub}_{\hat{\mathbb{G}}}, \mathsf{com}_{\hat{\mathbb{G}}}$	$\mathsf{base}_{\check{\mathbb{H}}}, \mathsf{pub}_{\check{\mathbb{H}}}, \mathsf{com}_{\check{\mathbb{H}}}$
$\mathrm{ME}_{\hat{\mathbb{G}}}$	$\mathsf{base}_{\hat{\mathbb{G}}}, \mathsf{pub}_{\hat{\mathbb{G}}}, \mathsf{com}_{\hat{\mathbb{G}}}$	$\mathsf{sca}_{\check{\mathbb{H}}}$
$\mathrm{ME}_{\check{\mathbb{H}}}$	$\mathsf{sca}_{\hat{\mathbb{G}}}$	$\mathsf{base}_{\check{\mathbb{H}}}, \mathsf{pub}_{\check{\mathbb{H}}}, \mathsf{com}_{\check{\mathbb{H}}}$
QE	$\mathsf{sca}_{\hat{\mathbb{G}}}$	$\mathsf{sca}_{\check{\mathbb{H}}}$

Figure 4.5: Possible GS commit labels for each equation type.

signer secret key sk (whose corresponding pk is included in $\mathcal{R}$). It outputs a new signature σ' with threshold $t + 1$.

- $\sigma' \leftarrow \mathsf{Extend}(m, \sigma, \{\mathsf{pk}_i\}_{i \in \mathcal{R}}, \{\mathsf{pk}_i\}_{i \in \mathcal{R}'})$: extends the signature σ with threshold t for the ring $\mathcal{R}$ into a new signature σ' with threshold t for the larger ring $\mathcal{R} \cup \mathcal{R}'$.

To formalize the properties of ETRS, we use the notion of ladder as in [5]. A ladder lad is a sequence of tuples $(\mathsf{action}, \mathsf{input})$, where action takes a value in the set $\{\mathsf{Sign}, \mathsf{Join}, \mathsf{Extend}\}$ and the value of input depends on the value of action. If $\mathsf{action} = \mathsf{Sign}$, then input is a pair $(\mathcal{R}, i)$, where $\mathcal{R}$ is the ring for the signature and i is the signer's identity. If $\mathsf{action} = \mathsf{Join}$, then input is an identifier i that identifies the signer that joins the signature. If $\mathsf{action} = \mathsf{Extend}$, then input is a

```
$\mathsf{Prove}(L, \Gamma, \{(l_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^m, \{(l_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^n)$
**if** $\boldsymbol{x} \in \hat{\mathbb{G}}^m$ define $\hat{C} = \boldsymbol{e}^\top \boldsymbol{x} + \hat{\boldsymbol{v}} \boldsymbol{r}_x + \hat{\boldsymbol{w}} \boldsymbol{s}_x$ **else if** $\boldsymbol{x} \in \mathbb{Z}_p^m$ define $\hat{C} = \hat{\boldsymbol{u}} \boldsymbol{x} + \hat{\boldsymbol{v}} \boldsymbol{r}_x$
**if** $\boldsymbol{y} \in \check{\mathbb{H}}^n$ define $\check{D} = \boldsymbol{e}^\top \boldsymbol{y} + \boldsymbol{r}_y \check{\boldsymbol{v}} + \boldsymbol{s}_y \check{\boldsymbol{w}}$ **else if** $\boldsymbol{y} \in \mathbb{Z}_p^n$ define $\check{D} = \check{\boldsymbol{u}} \boldsymbol{y} + \boldsymbol{r}_y \check{\boldsymbol{v}}$
Set $\alpha = \beta = \gamma = \delta = 0$
**if** $L = \text{PPE}$ $\alpha, \beta, \gamma, \delta \leftarrow\!\$\ \mathbb{Z}_p$
**if** $L = \text{ME}_{\hat{\mathbb{G}}}$ $\alpha, \beta \leftarrow\!\$\ \mathbb{Z}_p$
**if** $L = \text{ME}_{\check{\mathbb{H}}}$ $\alpha, \gamma \leftarrow\!\$\ \mathbb{Z}_p$
**if** $L = \text{QE}$ $\alpha \leftarrow\!\$\ \mathbb{Z}_p$
    $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \boldsymbol{r}_x \Gamma \check{D} + \alpha \check{\boldsymbol{v}} + \beta \check{\boldsymbol{w}}$      $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}} = (\hat{C} - \hat{\boldsymbol{v}} \boldsymbol{r}_x - \hat{\boldsymbol{w}} \boldsymbol{s}_x) \Gamma \boldsymbol{r}_y - \hat{\boldsymbol{v}} \alpha - \hat{\boldsymbol{w}} \gamma$
    $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \boldsymbol{s}_x \Gamma \check{D} + \gamma \check{\boldsymbol{v}} + \delta \check{\boldsymbol{w}}$      $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}} = (\hat{C} - \hat{\boldsymbol{v}} \boldsymbol{r}_x - \hat{\boldsymbol{w}} \boldsymbol{s}_x) \Gamma \boldsymbol{s}_y - \hat{\boldsymbol{v}} \beta - \hat{\boldsymbol{w}} \delta$
**return** $\boldsymbol{\pi} = (\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}})$
```

Figure 4.6: Prover algorithm of the GS proof system.

```
$\mathsf{PrVerify}(L, \Gamma, \{(l_{x_i}, \hat{\boldsymbol{c}}_i)\}_{i=1}^m, \{(l_{y_j}, \check{\boldsymbol{d}}_j)\}_{j=1}^n), \boldsymbol{\pi})$
Check that the equation has a valid format.
Check $\hat{C} = (\hat{\boldsymbol{c}}_1 \dots \hat{\boldsymbol{c}}_m) \in \hat{\mathbb{G}}^{2\times m}$ and $\check{D} = (\check{\boldsymbol{d}}_1 \dots \check{\boldsymbol{d}}_n)^\top \in \check{\mathbb{H}}^{n\times 2}$
Check $\boldsymbol{\pi} = (\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}) \in \check{\mathbb{H}}^{2\times 1} \times \check{\mathbb{H}}^{2\times 1} \times \hat{\mathbb{G}}^{1\times 2} \times \hat{\mathbb{G}}^{1\times 2}$
Check $\hat{C} \Gamma \check{D} = \hat{\boldsymbol{v}} \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} + \hat{\boldsymbol{w}} \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} + \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}} \check{\boldsymbol{v}} + \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}} \check{\boldsymbol{w}}$
**return** 1 if and only if all checks pass and 0 otherwise.
```

Figure 4.7: Verifier algorithm of the GS proof system.

ring $\mathcal{R}$ that is the ring to use to extend the previous ring. We notice that a ladder unequivocally determines a sequence of ETRS, each one with a specific ring and threshold value. In Fig. 4.11, the algorithm Proc is described. Proc takes as input a message, a ladder, and a corresponding list of keys, and outputs the sequence of all the signatures that correspond to each step of the ladder. It outputs $\perp$ whenever the ladder has an inconsistent sequence of actions or is incompatible with the list of keys provided in the input.

Definition 24 (Correctness for ETRS)**.** *For all $\lambda \in \mathbb{N}$, for any message $m \in \{0,1\}^*$, for any ladder* lad *of polynomial size identifying a*

```
$(\hat{C}', \check{D}', \pi') \leftarrow \mathsf{RandPr}(L, \Gamma, \{(l_{x_i}, \hat{\boldsymbol{c}}_i)\}_{i=1}^m, \{(l_{y_j}, \check{\boldsymbol{d}}_j)\}_{j=1}^n, \boldsymbol{\pi}; \boldsymbol{r})$
Parse $\boldsymbol{r} = (\boldsymbol{r}_x, \boldsymbol{s}_x, \boldsymbol{r}_y, \boldsymbol{s}_y)$
Define $\hat{C} = (\hat{\boldsymbol{c}}_1 \dots \hat{\boldsymbol{c}}_m) \in \hat{\mathbb{G}}^{2\times m}, \check{D} = (\check{\boldsymbol{d}}_1 \dots \check{\boldsymbol{d}}_n)^\top \in \check{\mathbb{H}}^{n\times 2}$
Parse $\boldsymbol{\pi} = (\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}})$
if $\boldsymbol{x} \in \hat{\mathbb{G}}^m$ define $\hat{C}' = \hat{C} + \hat{\boldsymbol{v}}\boldsymbol{r}_x + \hat{\boldsymbol{w}}\boldsymbol{s}_x$ else if $\boldsymbol{x} \in \mathbb{Z}_p^m$ define $\hat{C}' = \hat{C} + \hat{\boldsymbol{v}}\boldsymbol{r}_x$
if $\boldsymbol{y} \in \check{\mathbb{H}}^n$ define $\check{D}' = \check{D} + \boldsymbol{r}_y\check{\boldsymbol{v}} + \boldsymbol{s}_y\check{\boldsymbol{w}}$ else if $\boldsymbol{y} \in \mathbb{Z}_p^n$ define $\check{D}' = \check{D} + \boldsymbol{r}_y\check{\boldsymbol{v}}$
if $L = \mathrm{PPE}$ $\alpha, \beta, \gamma, \delta \leftarrow\$ \mathbb{Z}_p$
if $L = \mathrm{ME}_{\hat{\mathbb{G}}}$ $\alpha, \beta \leftarrow\$ \mathbb{Z}_p$
if $L = \mathrm{ME}_{\check{\mathbb{H}}}$ $\alpha, \gamma \leftarrow\$ \mathbb{Z}_p$
if $L = \mathrm{QE}$ $\alpha \leftarrow\$ \mathbb{Z}_p$
    $\check{\boldsymbol{\pi}}'_{\hat{\boldsymbol{v}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} + \boldsymbol{r}_x\Gamma\check{D}' + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$    $\hat{\boldsymbol{\pi}}'_{\check{\boldsymbol{v}}} = \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}} + \hat{C}\Gamma\boldsymbol{r}_y - \hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$
    $\check{\boldsymbol{\pi}}'_{\hat{\boldsymbol{w}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} + \boldsymbol{s}_x\Gamma\check{D}' + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}$    $\hat{\boldsymbol{\pi}}'_{\check{\boldsymbol{w}}} = \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}} + \hat{C}\Gamma\boldsymbol{s}_y - \hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$
return $(\hat{C}', \check{D}', \pi' = (\check{\boldsymbol{\pi}}'_{\hat{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}'_{\check{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}'_{\hat{\boldsymbol{w}}}, \hat{\boldsymbol{\pi}}'_{\check{\boldsymbol{w}}}))$
```

Figure 4.8: Proof re-randomization algorithm of the GS proof system.

ring $\mathcal{R}$, it holds that:

$$\Pr\left[\begin{array}{c|c} \left(\bigwedge_{j=1}^{\ell} \mathsf{Verify}(t, m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \sigma_j) = 1\right) & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ & \mathsf{L_{keys}} \leftarrow \{\mathsf{KeyGen}()\}_{i\in\mathcal{R}}; \\ \vee(\Sigma, t, \mathcal{R}) = \bot & (\Sigma, t, \mathcal{R}) \leftarrow \mathsf{Proc}(m, \mathsf{L_{keys}}, \mathsf{lad}); \\ & \{\sigma_1, \dots, \sigma_\ell\} = \Sigma \end{array}\right] = 1.$$

Definition 25 (Unforgeability for ETRS)**.** *An extendable threshold ring signature scheme* ETRS *is said to be unforgeable if for all PPT adversaries $\mathcal{A}$, the success probability in the experiment of Fig. 4.9 is*

$$\Pr\left[\mathsf{Exp}^{cmEUF}_{\mathcal{A},\mathsf{ETRS}}(\lambda) = win\right] \leq \mathsf{negl}(\lambda).$$

Definition 26 (Anonymity for ETRS)**.** *An extendable threshold ring signature scheme* ETRS *is said to provide anonymity if for all PPT adversaries $\mathcal{A}$, the success probability in the anonymous extendability experiment of Fig. 4.10 is* $\Pr\left[\mathsf{Exp}^{ANEXT}_{\mathcal{A},\mathsf{ETRS}}(\lambda) = win\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$*. In*

```
$\mathsf{Exp}^{\mathrm{cmEUF}}_{\mathcal{A},\mathsf{ETRS}}(\lambda)$
1 :  $\mathsf{L_{keys}}, \mathsf{L_{corr}}, \mathsf{L_{sign}}, \mathsf{L_{join}} \leftarrow \emptyset$
2 :  $\mathsf{pp} \leftarrow \mathsf{ETRS.Setup}(1^\lambda)$
3 :  $\mathsf{O} \leftarrow \{\mathsf{OSign}, \mathsf{OKey}, \mathsf{OCorr}, \mathsf{OJoin}\}$
4 :  $(t^*, m^*, \mathcal{R}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{O}}(pp)$
5 :  $q_1 \leftarrow |\{(m^*, \mathcal{R}, \cdot) \in \mathsf{L_{sign}} : \mathcal{R} \subseteq \mathcal{R}^*\}|$
6 :  $q_2 \leftarrow |\{(m^*, i, \cdot) \in \mathsf{L_{join}} : i \in \mathcal{R}^*\}|$
7 :  $q \leftarrow q_1 + q_2$
8 :  if $|\mathcal{R}^* \cap \mathsf{L_{corr}}| + q \geq t^*$
9 :    return $lose$
10 : if $\mathsf{Verify}(t^*, m^*, \{\mathsf{pk}_j\}_{j \in \mathcal{R}^*}, \sigma^*) = 0$
11 :   return $lose$
12 : return $win$

$\mathsf{OSign}(m, \mathcal{R}, i)$
1 :  if $(i \in \mathsf{L_{corr}} \vee i \notin \mathcal{R})$ return $\perp$
2 :  for $j \in \mathcal{R}$
3 :    if $(j, pk_j, \cdot) \notin \mathsf{L_{keys}}$
4 :      return $\perp$
5 :  $\sigma \leftarrow \mathsf{ETRS.Sign}(m, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \mathsf{sk}_i)$
6 :  $\mathsf{L_{sign}} \leftarrow \mathsf{L_{sign}} \cup \{(m, \mathcal{R}, i)\}$
7 :  return $\sigma$

$\mathsf{OKey}(i, \mathsf{pk})$
1 :  if $\mathsf{pk} = \perp$
2 :    $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{ETRS.KeyGen}()$
3 :    $\mathsf{L_{keys}} \leftarrow \mathsf{L_{keys}} \cup \{(i, \mathsf{pk}_i, \mathsf{sk}_i)\}$
4 :  else
5 :    $\mathsf{L_{corr}} \leftarrow \mathsf{L_{corr}} \cup \{i\}$
6 :    $\mathsf{pk}_i \leftarrow \mathsf{pk}$
7 :    $\mathsf{L_{keys}} \leftarrow \mathsf{L_{keys}} \cup \{(i, \mathsf{pk}_i, \perp)\}$
8 :  return $\mathsf{pk}_i$

$\mathsf{OCorr}(i)$
1 :  if $(i, \mathsf{pk}_i, \mathsf{sk}_i) \in \mathsf{L_{keys}} \wedge \mathsf{sk}_i \neq \perp$
2 :    $\mathsf{L_{corr}} \cup \{i\}$
3 :    return $(\mathsf{pk}_i, \mathsf{sk}_i)$
4 :  return $\perp$

$\mathsf{OJoin}(m, \mathcal{R}, i, \sigma)$
1 :  if $i \in \mathsf{L_{corr}}$ return $\perp$
2 :  for $j \in \mathcal{R}$
3 :    if $((j, pk_j, \cdot) \notin \mathsf{L_{keys}})$
4 :      return $\perp$
5 :  $\sigma' \leftarrow \mathsf{Join}(m, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \mathsf{sk}_i, \sigma)$
6 :  $\mathsf{L_{join}} \leftarrow \mathsf{L_{join}} \cup \{(m, i, \sigma)\}$
7 :  return $\sigma'$
```

Figure 4.9: Unforgeability game for ETRS (security experiment and oracles). This notion is reported from [5].

this experiment, the ladders submitted by $\mathcal{A}$ are said to be well-formed if all the actions in the ladder are pairwise of the same type, and they have the same ring as input.

Remarks on anonymity and unforgeability for ETRS. We modify our definition of anonymity for ETRS making it stronger w.r.t. the one in [5]. The difference is that the adversary now gets to see all the intermediate ETRS instead of just the final one (see lines 11 and 12 of Chal in Fig. 4.10). This modification enables count-me-in applications where partial signatures get publicly posted. In addition, in the experiment we add the checks of lines 15 and 17 to rule out a trivial attack inherent to any ETRS. Indeed, since the Join operation cannot increase the threshold of an ETRS when using a secret key that was already used before, $\mathcal{A}$ could use this fact to distinguish between the ladders.

The Combine algorithm is introduced in [5] as a procedure to combine together two signatures on the same message with two different (not necessarily disjoint) rings. The output is a signature having as ring the union of the two rings and as threshold the cardinality of the union of the signers sets of the starting signatures. The Combine algorithm can be run without knowing any secret key. In [5], the authors showed that the Join operation can be obtained as the concatenation of the Sign operation and the Combine operation. In order to avoid the same attack described before, the checks in lines 11 and 13 of the experiment of Fig. 4.10 are needed. We notice that our ETRS only provides a weaker form of Combine in which the starting rings are disjoint (cfr., Sec. 4.2.8). A similar discussion holds for lines $5-8$ of the unforgeability experiment in Fig. 4.9. In particular, they rule out trivial attacks due to $\mathcal{A}$ asking too many sign, join, or corruption queries.

Fellow signer anonymity. We also define a stronger version of anonymity called fellow signer anonymity. This game models the requirement that even a signer cannot determine any of the other signers by just looking at all the signatures that were produced. It is straightforward to notice that fellow signer anonymity implies anonymity for ETRS.

Definition 27 (Fellow Signer Anonymity for ETRS). *An extendable threshold ring signature scheme* ETRS *is said to provide fellow signer anonymity if for all PPT adversaries $\mathcal{A}$, the success probability in the*

```
$\mathsf{Exp}^{\mathrm{ANEXT}}_{\mathcal{A},\mathsf{ETRS}}(\lambda)$
$b \leftarrow\!\$ \{0,1\}$
$\mathsf{L_{keys}}, \mathsf{L_{corr}}, \mathsf{L_{sign}}, \mathsf{L_{join}} \leftarrow \emptyset$
$pp \leftarrow \mathsf{ETRS.Setup}(1^\lambda)$
$(m^*, \mathsf{lad}_0^*, \mathsf{lad}_1^*) \leftarrow \mathcal{A}^{\mathsf{O}}(pp)$
$\Sigma \leftarrow \mathsf{Chal}_b(m^*, \mathsf{lad}_0^*, \mathsf{lad}_1^*)$
$b^* \leftarrow \mathcal{A}^{\mathsf{O}}(\Sigma)$
$\textbf{if } \exists i \in \mathsf{lad}_0^*.\mathcal{S} \ s.t.\ i \in \mathsf{L_{corr}}$
  $\textbf{return } lose$
$\textbf{if } \exists i \in \mathsf{lad}_1^*.\mathcal{S} \ s.t.\ i \in \mathsf{L_{corr}}$
  $\textbf{return } lose$
$\textbf{for } i \in \mathsf{lad}_0^*.\mathcal{S} \cup \mathsf{lad}_1^*.\mathcal{S}$
  $\textbf{if } \exists (m^*, \cdot, i) \in \mathsf{L_{sign}}$
    $\textbf{return } lose$
  $\textbf{if } \exists (m^*, i, \cdot) \in \mathsf{L_{join}}$
    $\textbf{return } lose$
$\textbf{if } b^* \neq b \textbf{ return } lose$
$\textbf{return } win$
```

```
$\mathsf{Chal}_b(m^*, \mathsf{lad}_0^*, \mathsf{lad}_1^*)$
$\textbf{if } (\mathsf{lad}_0^*, \mathsf{lad}_1^*)$ is not well-formed
  $\textbf{return } \perp$
$\textbf{if } \exists i \in \mathsf{lad}_0^*.\mathcal{S} \ s.t.\ i \in \mathsf{L_{corr}}$
  $\textbf{return } \perp$
$\textbf{if } \exists i \in \mathsf{lad}_1^*.\mathcal{S} \ s.t.\ i \in \mathsf{L_{corr}}$
  $\textbf{return } \perp$
$val_0 \leftarrow \mathsf{Proc}(m^*, \mathsf{L_{keys}}, \mathsf{lad}_0^*)$
$val_1 \leftarrow \mathsf{Proc}(m^*, \mathsf{L_{keys}}, \mathsf{lad}_1^*)$
$\textbf{if } val_0 = \perp \vee val_1 = \perp$
  $\textbf{return } \perp$
Parse $val_0$ as $(\Sigma_0, t_0, \mathcal{R}_0)$
Parse $val_1$ as $(\Sigma_1, t_1, \mathcal{R}_1)$
$\Sigma \leftarrow \Sigma_b$
$\textbf{return } \Sigma$
```

Figure 4.10: Anonymous extendability game. We use $\mathsf{lad}.\mathcal{S}$ to indicate the set of signers of a ladder lad. $\mathcal{A}$ has access to oracles $\mathsf{O} \leftarrow \{\mathsf{OSign}, \mathsf{OKey}, \mathsf{OCorr}, \mathsf{OJoin}\}$. We propose a stronger notion compared to [5]. Indeed, in our definition the adversary gets to see all the intermediate signatures instead of only the final ETRS.

experiment of Fig. 4.12 is $\Pr\big[\mathsf{Exp}^{ANFS}_{\mathcal{A},\mathsf{ETRS}}(\lambda) = win\big] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

4.2.7 Extendable Non-interactive Witness Indistinguishable Proof of Knowledge

Given an NP language $\mathcal{L}$ with associated poly-time relation $\mathcal{R}_{\mathcal{L}}$, we define the related threshold relation $\mathcal{R}_t$ as follows.

```
$\mathsf{Proc}(m, \mathsf{L_{keys}}, \mathsf{lad})$
$\Sigma \leftarrow \emptyset, t = 0$
Parse $\mathsf{lad}$ as $((\mathsf{action}^1, \mathsf{input}^1), \ldots, (\mathsf{action}^l, \mathsf{input}^l))$
if $\mathsf{action}^1 \neq \mathsf{Sign}$ return $\perp$
else
    Parse $\mathsf{input}^1$ as $(\mathcal{R}^1, i^1)$
    for $j \in \mathcal{R}^1$ if $(j, \mathsf{pk}_j, \cdot) \notin \mathsf{L_{keys}}$ return $\perp$
    if $sk_{i^1} = \perp \vee i^1 \notin \mathcal{R}^1$ return $\perp$
    $\mathcal{R} \leftarrow \mathcal{R}^1, \mathcal{S} \leftarrow \{i^1\}$
    $\sigma \leftarrow \mathsf{Sign}(m, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \mathsf{sk}_{i^1}), \Sigma \leftarrow \Sigma \cup \{\sigma\}$
for $l' \in [2, \ldots, l]$
    if $\mathsf{action}^{l'} = \mathsf{Sign}$ return $\perp$
    else
        if $\mathsf{action}^{l'} = \mathsf{Join}$ parse $\mathsf{input}^{l'}$ as $(i^{l'})$
            if $i^{l'} \notin \mathcal{R} \vee i^{l'} \in \mathcal{S}$ return $\perp$
            $\sigma \leftarrow \mathsf{Join}(m, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \mathsf{sk}_i^{l'}, \sigma)$
            $\Sigma \leftarrow \Sigma \cup \{\sigma\}, \mathcal{S} \leftarrow \mathcal{S} \cup \{i^{l'}\}, t = t + 1$
        if $\mathsf{action}^{l'} = \mathsf{Extend}$ parse $\mathsf{input}^{l'}$ as $(\mathcal{R}^{l'})$
            for $j \in \mathcal{R}^{l'}$ if $(j, \mathsf{pk}_j, \cdot) \notin \mathsf{L_{keys}}$ return $\perp$
            $\sigma \leftarrow \mathsf{Extend}(m, \sigma, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \{\mathsf{pk}_j\}_{j \in \mathcal{R}^{l'}})$
            $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}^{l'}, \Sigma \leftarrow \Sigma \cup \{\sigma\}$
        else return $\perp$
return $(\Sigma, t, \mathcal{R})$
```

Figure 4.11: Process algorithm for ETRS.

$$\mathcal{R}_t = \{(x = (k, x_1, \ldots, x_n), w = ((w_1, \alpha_1), \ldots, (w_k, \alpha_k)))| \\ 1 \leq \alpha_1 < \ldots < \alpha_k \leq n \wedge \forall\ j \in [k] : (x_{\alpha_j}, w_j) \in \mathcal{R}_{\mathcal{L}}\}.$$

An extendable non-interactive proof system for a threshold relation

```
$\mathsf{Exp}^{\mathrm{ANFS}}_{\mathcal{A},\mathsf{ETRS}}(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathsf{L}_{\mathsf{keys}}, \mathsf{L}_{\mathsf{corr}}, \mathsf{L}_{\mathsf{sign}}, \mathsf{L}_{\mathsf{join}} \leftarrow \emptyset$
$\mathsf{p} \leftarrow \mathsf{ETRS.Setup}(1^\lambda)$
$(m^*, \mathsf{lad}^*, i^*, j^*) \leftarrow \mathcal{A}^{\mathsf{O}}(pp)$
$\Sigma \leftarrow \mathsf{Chal}_b(m^*, \mathsf{lad}^*, i^*, j^*)$
$b^* \leftarrow \mathcal{A}^{\mathsf{O}}(\Sigma)$
$\mathbf{if}\ i^* \in \mathsf{L}_{\mathsf{corr}} \vee j^* \in \mathsf{L}_{\mathsf{corr}}$
  $\mathbf{return}\ lose$
$\mathbf{if}\ \exists\,(m^*, \cdot, i^*) \in \mathsf{L}_{\mathsf{sign}}$
  $\mathbf{return}\ lose$
$\mathbf{if}\ \exists(m^*, \cdot, j^*) \in \mathsf{L}_{\mathsf{sign}}$
  $\mathbf{return}\ lose$
$\mathbf{if}\ \exists\,(m^*, i^*, \cdot) \in \mathsf{L}_{\mathsf{join}}$
  $\mathbf{return}\ lose$
$\mathbf{if}\ \exists(m^*, j^*, \cdot) \in \mathsf{L}_{\mathsf{join}}$
  $\mathbf{return}\ lose$
$\mathbf{if}\ b^* \neq b$
  $\mathbf{return}\ lose$
$\mathbf{return}\ win$
```

```
$\mathsf{Chal}_b(m^*, \mathsf{lad}^*, i^*, j^*)$
$\mathbf{if}\ i^* \in \mathsf{L}_{\mathsf{corr}} \vee j^* \in \mathsf{L}_{\mathsf{corr}}$
  $\mathbf{return}\ \bot$
$\mathsf{lad}^*.add((\mathsf{Extend}, \{i^*\}))$
$\mathsf{lad}^*.add((\mathsf{Extend}, \{j^*\}))$
$\mathbf{if}\ b = 0$
  $\mathsf{lad}^*.add((\mathsf{Join}, i^*))$
$\mathbf{if}\ b = 1$
  $\mathsf{lad}^*.add((\mathsf{Join}, j^*))$
$val \leftarrow \mathsf{Proc}(m^*, \mathsf{L}_{\mathsf{keys}}, \mathsf{lad}^*)$
$\mathbf{if}\ val = \bot$
  $\mathbf{return}\ \bot$
$\mathbf{else}$
  Parse $val$ as $(\Sigma, t, \mathcal{R})$
$\mathbf{return}\ \Sigma$
```

Figure 4.12: Fellow signer anonymity game. We use $\mathsf{lad}.\mathcal{S}$ to indicate the set of signers of a ladder lad and $\mathsf{lad}.add$ to indicate that we are adding the pair $(\mathsf{action}, \mathsf{input})$ as the last element of the ladder. $\mathcal{A}$ has access to oracles $\mathsf{O} \leftarrow \{\mathsf{OSign}, \mathsf{OKey}, \mathsf{OCorr}, \mathsf{OJoin}\}$.

$\mathcal{R}_t$ consists of the following PPT algorithms. The group key $\mathsf{gk} \leftarrow \mathsf{Gen}(1^\lambda)$ is considered as an implicit input to all algorithms:

- $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$: on input the group key gk, output a uniformly random[22] common reference string $\mathsf{crs} \in \{0,1\}^\lambda$.

[22]Here we are also assuming that the crs is uniformly random since it is needed by our ETRS construction.

- $(\Pi, (\mathsf{aux}_1, \ldots, \mathsf{aux}_n)) \leftarrow \mathsf{Prove}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)))$: on input $((k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k))) \in \mathcal{R}_t$, output a proof Π and auxiliary values $(\mathsf{aux}_1, \ldots, \mathsf{aux}_n)$. The auxiliary value aux_i is used later on to perform an add operation using a witness for a not previously used statement x_i.

- $0/1 \leftarrow \mathsf{PrVerify}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi)$: on input statement $(k, x_1, \ldots, x_n)$, and a proof Π, output 1 to accept and 0 to reject.

- $(\Pi', \mathsf{aux}_{n+1}) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, (k, x_1, \ldots, x_n), x_{n+1}, \Pi)$: on input statements $(k, x_1, \ldots, x_n)$, x_{n+1}, and a proof Π for $(k, x_1, \ldots, x_n) \in \mathcal{L}_t$, output an updated proof Π' for $(k, x_1, \ldots, x_n, x_{n+1}) \in \mathcal{L}_t$, and additional auxiliary value aux_{n+1}. The auxiliary value aux_{n+1} is used later on to perform an add operation using a witness for x_{n+1}.

- $(\Pi', \mathsf{aux}'_\alpha) \leftarrow \mathsf{PrAdd}(\mathsf{crs}, (k, x_1, \ldots, x_n), (w, \alpha), \mathsf{aux}, \Pi)$: on input statement $(k, x_1, \ldots, x_n)$, witness (w, α), auxiliary value aux, and proof Π for $(k, x_1, \ldots, x_n) \in \mathcal{L}_t$, output an updated proof Π' for $(k+1, x_1, \ldots, x_n) \in \mathcal{L}_t$, and updated auxiliary value aux'_α.

- $(\Pi', r = (r_1, \ldots, r_n)) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi)$: on input statement x and proof Π for $x \in \mathcal{L}_t$, output a re-randomized proof Π' and update randomness r_i (related to auxiliary value aux_i) with $i \in [n]$.

- $\mathsf{aux}'_i \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{aux}_i, r_i)$: on input auxiliary value aux_i, and update randomness r_i, output updated auxiliary value aux'_i. $\mathsf{AuxUpdate}$ is used to update the auxiliary values after a proof has been re-randomized. The used input randomness is the one given in output by RandPr. To simplify the notation, we write $\mathsf{AUX}' \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}, r)$ to indicate that a list of auxiliary values is updated by appropriately parsing AUX and r and running the update operation on each element of the list.

- $0/1 \leftarrow \mathsf{AuxVerify}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)), (\mathsf{aux}_1, \ldots, \mathsf{aux}_n), \Pi)$: on input statement $(k, x_1, \ldots, x_n)$, witness $((w_1, \alpha_1) \ldots, (w_k, \alpha_k))$, auxiliary values $(\mathsf{aux}_1, \ldots, \mathsf{aux}_n)$, and proof Π, output 1 if the auxiliary values are consistent with the statement, the proof, and the witness. Return 0 otherwise. If $\mathsf{AuxUpdate}$ returns 1, we are guaranteed that the subsequent extend/add operations can be correctly executed[23].

An extendable non-interactive proof system is said to be an extendable non-interactive witness indistinguishable (ENIWI) proof of knowledge if it satisfies adaptive extractable soundness (Def. 17) and the following properties.

Definition 28 (Completeness). *An extendable non-interactive proof system for $\mathcal{R}_t$ is complete if $\forall\lambda \in \mathbb{N}$, $\mathsf{gk} \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$, $(x, w) \in \mathcal{R}_t$, and $(\Pi, \mathsf{AUX}) \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$ it holds that*

$$\Pr[\mathsf{PrVerify}(\mathsf{crs}, x, \Pi) = 1 \wedge \mathsf{AuxVerify}(\mathsf{crs}, x, w, \mathsf{AUX}, \Pi) = 1] = 1$$

Definition 29 (Transformation Completeness). *An extendable non-interactive proof system for $\mathcal{R}_t$ is transformation complete if $\forall\lambda \in \mathbb{N}$, $\mathsf{gk} \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$, $(x, w) \in \mathcal{R}_t$, and (Π, AUX) such that $\mathsf{PrVerify}(\mathsf{crs}, x, \Pi) = 1$ and $\mathsf{AuxVerify}(\mathsf{crs}, x, w, \mathsf{AUX}, \Pi) = 1$ the following holds with probability 1:*

- $\mathsf{AuxVerify}(\mathsf{crs}, x, w, \mathsf{AUX}', \Pi') = 1$, *where* $(\Pi', r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \Pi)$ *and* $\mathsf{AUX}' \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}, r)$.
- *Parse x as $(k, x_1, \dots, x_n)$ and w as $((w_1, \alpha_1)\dots, (w_k, \alpha_k))$.* $(\Pi', \mathsf{aux}') \leftarrow \mathsf{PrAdd}(\mathsf{crs}, x, (w', \alpha'), \mathsf{aux}, \Pi)$, *modify* AUX *replacing* $\mathsf{aux}_{\alpha'}$ *with* aux'.

  If $\alpha' \notin \{\alpha_1, \dots \alpha_k\}$, $(x_{\alpha'}, w') \in \mathcal{R}_{\mathcal{L}}$, then $\mathsf{PrVerify}(\mathsf{crs}, (k{+}1, x_1, \dots, x_n), \Pi') = 1$, *and* $\mathsf{AuxVerify}(\mathsf{crs}, (k{+}1, x_1, \dots, x_n), ((w_1, \alpha_1)\dots, (w_k, \alpha_k), (w', \alpha')), \mathsf{AUX}, \Pi') = 1$.
- $(\Pi', \mathsf{aux}_{n+1}) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, x, x_{n+1}, \Pi)$, *modify* AUX *adding auxiliary value* aux_{n+1}. *Then,* $\mathsf{PrVerify}(\mathsf{crs}, (k, x_1, \dots, x_{n+1}), \Pi') = 1$, *and* $\mathsf{AuxVerify}(\mathsf{crs}, (k, x_1, \dots, x_{n+1}), w, \mathsf{AUX}, \Pi') = 1$.

Definition 30 (Re-Randomizable Addition). *Consider the following experiment:*

[23] We introduce $\mathsf{AuxVerify}$ merely as an internal utility to simplify the description of our definitions.

- $\mathsf{gk} \leftarrow \mathsf{Gen}(1^\lambda)$
- $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$
- $(x, w, \Pi^*, \mathsf{AUX}^*) \leftarrow \mathcal{A}(\mathsf{crs})$
- *Parse* x *as* $(k, x_1, \ldots, x_n)$ *and* w *as* $((w_1, \alpha_1) \ldots, (w_k, \alpha_k))$
- *If* $(x, w) \notin \mathcal{R}_t$ *or* $\mathsf{PrVerify}(\mathsf{crs}, (k-1, x_1, \ldots, x_n), \Pi^*) = 0$ *or* $\mathsf{AuxVerify}(\mathsf{crs}, (k-1, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_{k-1}, \alpha_{k-1})), \mathsf{AUX}^*, \Pi^*) = 0$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\!\!\$\ \{0,1\}$ *and do the following:*
  - *If* $b = 0$, $(\Pi_0, \mathsf{AUX}_0) \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$; $(\Pi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \Pi_0), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}_0, r)$
  - *If* $b = 1$, $(\Pi_1, \mathsf{aux}^*) \leftarrow \mathsf{PrAdd}(\mathsf{crs}, x, (w_k, \alpha_k), \mathsf{AUX}^*, \Pi^*)$. *Replace in* AUX^* *the value* aux_{α_k} *with* aux^*. $(\Pi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \Pi_1), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}^*, r)$
- $b' \leftarrow \mathcal{A}(\Pi, \mathsf{AUX})$

We say that the proof system has re-randomizable addition if for every PPT $\mathcal{A}$*, there exists a negligible function* $\nu(\cdot)$*, such that* $\Pr[b = b'] \leq 1/2 + \nu(\lambda)$.

Definition 31 (Re-Randomizable Extension). *Consider the following experiment:*

- $\mathsf{gk} \leftarrow \mathsf{Gen}(1^\lambda)$
- $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$
- $(x, w, x_n, \Pi^*, \mathsf{AUX}^*) \leftarrow \mathcal{A}(\mathsf{crs})$
- *Parse* x *as* $(k, x_1, \ldots, x_{n-1})$
- *If* $(x, w) \notin \mathcal{R}_t$ *or* $\mathsf{PrVerify}(\mathsf{crs}, x, \Pi^*) = 0$ *or* $\mathsf{AuxVerify}(\mathsf{crs}, x, w, \mathsf{AUX}^*, \Pi^*) = 0$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\!\!\$\ \{0,1\}$ *and do the following:*

  - *If* $b = 0$ $(\Pi_0, \mathsf{AUX}_0) \leftarrow \mathsf{Prove}(\mathsf{crs}, (k, x_1, \ldots, x_n), w)$*;* $(\Pi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi_0), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}_0, r)$
  - *If* $b = 1$ $(\Pi_1, \mathsf{aux}^*) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, x, x_n, \Pi^*)$*. Append the value* aux^* *to* AUX^**.* $(\Pi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi_1)$, $\mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}^*, r)$

- $b' \leftarrow \mathcal{A}(\Pi, \mathsf{AUX})$

We say that the proof system has re-randomizable extension if for every PPT $\mathcal{A}$*, there exists a negligible function* $\nu(\cdot)$*, such that* $\Pr[b = b'] \leq 1/2 + \nu(\lambda)$*.*

Definition 32 (Extended Witness Indistinguishability)**.** *Consider the following experiment.*

- $\mathsf{gk} \leftarrow \mathsf{Gen}(1^\lambda)$
- $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$
- $(x, w^0, w^1) \leftarrow \mathcal{A}(\mathsf{crs})$
- *Parse* x *as* $(k, x_1, \ldots, x_n)$*,* w^i *as* $((w_1^i, \alpha_1^i) \ldots, (w_k^i, \alpha_k^i))$*, for* $i \in \{0, 1\}$
- *If* $(x, w^0) \notin \mathcal{R}_t$ *or* $(x, w^1) \notin \mathcal{R}_t$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\!\!\$\ \{0, 1\}$ *and do the following:*
  - $(\Pi, (\mathsf{aux}_1, \ldots, \mathsf{aux}_n)) \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w_b)$*.*
  - *Set* $I_0 = \{\alpha_1^0, \ldots, \alpha_k^0\}$*,* $I_1 = \{\alpha_1^1, \ldots, \alpha_k^1\}$*,* $I = I_0 \cap I_1$*,* $S = ([n] \setminus (I_0 \cup I_1)) \cup I$*, and* $\mathsf{AUX} = \{\mathsf{aux}_i\}_{i \in S}$*.*
- $b' \leftarrow \mathcal{A}(\Pi, \mathsf{AUX})$

We say that the proof system has extended witness indistinguishability (EWI) *if for every PPT* $\mathcal{A}$*, there exists a negligible function* $\nu(\cdot)$*, such that* $\Pr[b = b'] \leq 1/2 + \nu(\lambda)$*.*

4.2.8 Our Extendable Threshold Ring Signature

In Fig. 4.13 and Fig. 4.14, we show our ETRS from an ENIWI PoK ENIWI for a *hard* relation $\mathcal{R}_{\mathcal{L}}$, and an IND-CPA public key encryption scheme PKE which is homomorphic w.r.t. $\mathsf{ENIWI.AuxUpdate}$. By *hard* relation we mean that a PPT $\mathcal{A}$ who is given $x \in \mathcal{L}$, has negligible probability of providing a witness w such $(x, w) \in \mathcal{R}_{\mathcal{L}}$. We also require that $\mathcal{R}_{\mathcal{L}}$ is public coin samplable, meaning that it is possible to efficiently sample random $x \in \mathcal{L}$. We omit the Setup algorithm from the description since it simply runs the setup algorithm of PKE and samples a hash function mapping arbitrary strings into elements in the correct space[24].

Instantiating our ETRS. We work over a bilinear group $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$ for which the SXDH assumption is believed to hold. In Sec. 4.2.10, we show an ENIWI PoK having as base relation pairing product equations in which all the variables are elements of $\check{\mathbb{H}}$, public constants are either paired with secret values or with $\check{h}$, and the target element is $0_{\mathbb{T}}$. In particular, we can use as base relation the following: $\mathcal{R}_{\mathcal{L}} = \{(x = (\hat{a}, \hat{b}, \check{h}), w = \check{b}' | \hat{a} \cdot \check{h} + \hat{b} \cdot \check{b}' = 0_{\mathbb{T}}\}$. In Lem. 5, we prove that this is a hard relation under the DDH assumption in $\hat{\mathbb{G}}$. Additionally, since in our ENIWI $\mathsf{AuxUpdate}$ simply consists of applying the group operation between two elements of $\check{\mathbb{H}}$, we can use ElGamal instantiated in $\check{\mathbb{H}}$ as public key encryption scheme.

Remark on malicious extenders. As in [5], we do not consider security definitions accounting for malicious signers that try to prevent future signers from joining the signature. For example, in our construction a malicious extender could just encrypt a wrong auxiliary value. An approach that could be investigated to tackle this issue is adding a NIZK proving that the content of the encrypted auxiliary values is s.t. $\mathsf{AuxVerify} = 1$. Such NIZK would need to be malleable so that it could be updated after every re-randomization step, as well as whenever the signature is extended.

[24]Our ENIWI PoK is based on GS, so we need a cryptographic hash function that allows to hash directly to both the source groups of the pairing group. See [61] for more details.

On combining signatures. One might wonder if concrete instantiations of our ETRS could also support the Combine operation as described in [5]. Whenever there is a shared public key (i.e., statement) in two ETRS, such signatures cannot be combined. Indeed, consider the case of two proofs over the same ring where there is a common base statement for which a corresponding witness was used in both proofs. Then, the combined proof should not have a resulting threshold that counts it twice. This means that the output of Combine would be different depending on whether two NIWI proofs on the same statement used the same witness or not. This is in clear contradiction with the witness indistinguishability property. On the other hand, the above observation does not exclude the possibility of having a weaker form of Combine where the starting signatures are constrained to have disjoint rings. Indeed, our instantiation of Sec. 4.2.13 could be easily modified to support the corresponding Combine operation. Such operation exploits basically the same technique of the extend operation, and thus we omit its description.

```
$\mathsf{Sign}(m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{sk})$
  $\mathsf{A} \leftarrow \emptyset$
  $(\mathsf{crs}, \mathsf{pk_O} = (x_1, \mathsf{pk_e^1})) \leftarrow \mathsf{H}(m)$
  $(\mathsf{pk}_2, \ldots, \mathsf{pk}_{n+1}) \lll \{\mathsf{pk}_i\}_{i\in\mathcal{R}}$
  $\textbf{for } i \in [n+1]$
    $(x_i, \mathsf{pk_e^i}) \lll \mathsf{pk}_i$
    $(w, \mathsf{sk_e}) \lll \mathsf{sk}$
  $\textbf{if } \nexists x_j, \text{ s.t. } (x_j, w) \in \mathcal{R_L}$
    $\textbf{return } \perp$
  Let $x = (1, x_1, \ldots, x_n, x_{n+1})$
  $(\Pi, \mathsf{AUX}) \leftarrow \mathsf{ENIWI.Prove}(x, (w, j))$
  $\textbf{for } i \in [n+1]$
    $\textbf{if } i = j \lor i = 1$
      $\mathsf{a} \leftarrow \mathsf{PKE.Enc}(\perp, \mathsf{pk_e^j})$
    $\textbf{else}$
      $\mathsf{a} \leftarrow \mathsf{PKE.Enc}(\mathsf{AUX}[i], \mathsf{pk_e^i})$
    $\mathsf{A} \leftarrow \mathsf{A} \cup \mathsf{a}$
  $\textbf{return } \sigma = (1, \Pi, \mathsf{A})$

$\mathsf{KeyGen}()$
  $(\mathsf{pk_e}, \mathsf{sk_e}) \leftarrow \mathsf{PKE.KeyGen}()$
  Sample $(x, w) \in \mathcal{R_L}$
  $(\mathsf{pk} = (x, \mathsf{pk_e}), \mathsf{sk} = (w, \mathsf{sk_e}))$
  $\textbf{return } (\mathsf{pk}, \mathsf{sk})$

$\mathsf{Verify}(t, m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \sigma)$
  $(\mathsf{crs}, \mathsf{pk_O} = (x_1, \mathsf{pk_e^1})) \leftarrow \mathsf{H}(m)$
  $(\mathsf{pk}_2, \ldots, \mathsf{pk}_{n+1}) \lll \{\mathsf{pk}_i\}_{i\in\mathcal{R}}$
  $\textbf{for } i \in [n+1]$
    Parse $\mathsf{pk}_i = (x_i, \mathsf{pk_e^i})$
  Parse $\sigma = (k, \Pi, \mathsf{A})$
  Let $x = (k, x_1, \ldots, x_{n+1})$
  $\textbf{if } k < t$
    $\textbf{return } 0$
  $\textbf{else}$
    $\textbf{return } \mathsf{ENIWI.PrVerify}(x, \Pi)$
```

Figure 4.13: ETRS from ENIWI PoK and IND-CPA homomorphic PKE. For space reasons, we omit crs from the input of the ENIWI algorithms and consider it as an implicit input. We use AUX[i] to indicate the i-th element of list AUX. The symbol $\lll$ indicates parsing of data. In this figure, we report the Sign, KeyGen, and Verify algorithms.

```
$\mathsf{Extend}(m, \sigma, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{pk}^*)$
1 :  $\textbf{if } \mathsf{pk}^* \in \{\mathsf{pk}_i\}_{i\in\mathcal{R}} \textbf{ return } \perp$
2 :  $(\mathsf{pk}_{\mathsf{O}} = (x_1, \mathsf{pk}_{\mathsf{e}}^1)) \leftarrow \mathsf{H}(m)$
3 :  $(\mathsf{pk}_2, \dots, \mathsf{pk}_{n+1}) \lll \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, (x_{n+2}, \mathsf{pk}_{\mathsf{e}}^{n+2}) \lll \mathsf{pk}^*, (k, \Pi, \mathsf{A}) \lll \sigma$
4 :  $(x_i, \mathsf{pk}_{\mathsf{e}}^i) \lll \mathsf{pk}_i$ for $i \in [n+1]$
5 :  Let $x = (k, x_1, \dots, x_{n+1})$
6 :  $(\Pi, \mathsf{aux}) \leftarrow \mathsf{ENIWI.PrExtend}(x, x_{n+2}, \Pi)\mathsf{a} \leftarrow$
7 :  $\mathsf{PKE.Enc}(\mathsf{aux}, \mathsf{pk}_{\mathsf{e}}^{n+2})$
8 :  $\mathsf{A} \leftarrow \mathsf{A} \cup \mathsf{a}$, Let $\bar{x} = (k, x_1, \dots, x_{n+2})$
9 :  $(\Pi, r_1, \dots, r_{n+2}) \leftarrow \mathsf{ENIWI.RandPr}(\bar{x}, \Pi)$
10 : $\textbf{for } \mathsf{a}_i \in \mathsf{A}$
11 :     $\mathsf{a}_i \leftarrow \mathsf{PKE.Eval}(\mathsf{a}_i, r_i, \mathsf{pk}_{\mathsf{e}}^i)$
12 : $\textbf{return } \sigma = (k, \Pi, \mathsf{A})$

$\mathsf{Join}(m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{sk}, \sigma)$
1 :  $(\mathsf{pk}_{\mathsf{O}} = (x_1, \mathsf{pk}_{\mathsf{e}}^1)) \leftarrow \mathsf{H}(m)$
2 :  $(\mathsf{pk}_2, \dots, \mathsf{pk}_{n+1}) \lll \{\mathsf{pk}_i\}_{i\in\mathcal{R}}$
3 :  $(x_i, \mathsf{pk}_{\mathsf{e}}^i) \lll \mathsf{pk}_i$ for $i \in [n+1]$
4 :  $(w, \mathsf{sk}_{\mathsf{e}}) \lll \mathsf{sk}$
5 :  $\textbf{if } \nexists x_j, j \in [n+1]$ s.t $(x_j, w) \in \mathcal{R}_{\mathcal{L}}$
6 :     $\textbf{return } \perp$
7 :  $(k, \Pi, \mathsf{A}) \lll \sigma, (\mathsf{a}_1, \dots, \mathsf{a}_{n+1}) \lll \mathsf{A}, (w, \mathsf{sk}_{\mathsf{e}}) \lll \mathsf{sk}$
8 :  $\mathsf{aux} \leftarrow \mathsf{PKE.Dec}(\mathsf{a}_j, \mathsf{sk}_{\mathsf{e}})$, Let $x = (k, x_1, \dots, x_{n+1})$
9 :  $(\Pi, \mathsf{aux}'_j) \leftarrow \mathsf{ENIWI.PrAdd}(x, (w, j), \mathsf{aux}, \Pi)$
10 : Set $\mathsf{a}_j \in \mathsf{A}$ as $\mathsf{a}_j \leftarrow \mathsf{PKE.Enc}(\perp, \mathsf{pk}_{\mathsf{e}}^j), k \leftarrow k + 1$
11 : $(\Pi, r_1, \dots, r_{n+1}) \leftarrow \mathsf{ENIWI.RandPr}(x, \Pi)$
12 : $\textbf{for } \mathsf{a}_i \in \mathsf{A}$
13 :    $\mathsf{a}_i \leftarrow \mathsf{PKE.Eval}(\mathsf{a}_i, r_i, \mathsf{pk}_{\mathsf{e}}^i)$
14 : $\textbf{return } \sigma = (k, \Pi, \mathsf{A})$
```

Figure 4.14: ETRS from ENIWI PoK and IND-CPA homomorphic PKE. In this figure, we report the Extend and Join algorithms.

4.2.9 Security of Our Extendable Threshold Ring Signature

We now prove the following theorem regarding the security of our ETRS.

Theorem 2. *Let* ENIWI *be an extendable non-interactive witness indistinguishable proof of knowledge for an hard relation* $\mathcal{R}_{\mathcal{L}}$*, and* PKE *be an* IND-CPA *public key encryption scheme which is homomorphic w.r.t.* ENIWI.AuxUpdate*, then the scheme reported in Fig. 4.13 and Fig. 4.14 is an extendable threshold ring signature scheme.*

We will prove Thm. 2 using Lem. 6 to prove that the signature scheme described in Fig. 4.13 and Fig. 4.14 is unforgeable and Lem. 7 to prove that such signature scheme is anonymous. For the sake of clarity, we first give proof sketches. After that, we provide complete proofs with detailed reductions.

Lemma 6. *The signature scheme described in Fig. 4.13 and Fig. 4.14 is unforgeable according to Def. 25.*

Proof sketch. The basic idea of the proof is to turn an adversary $\mathcal{A}$ breaking the unforgeability with non-negligible probability into another adversary $\mathcal{B}$ that extracts a witness for an instance $x \in \mathcal{L}$ of the hard relation, which is sampled by a challenger $\mathcal{C}$. In order to build this reduction, we need to show how to simulate all the oracle queries of $\mathcal{A}$ during the game. We do this by showing a series of hybrid games, starting from the game described in Fig. 4.9.

The first change consists into replying to Join queries by computing every time a new proof from scratch using ENIWI.Prove, instead of updating the current proof using PrAdd. This change is not detected by $\mathcal{A}$ thanks to the re-randomizable addition of the ENIWI.

The second change is that $\mathcal{B}$ can guess j^*, that is the index of the random oracle query in which $\mathcal{A}$ will query the message used in the forgery, and i^*, that is the index of a "new" signer used to create the forgery for m_{j^*}. We notice that, by the rules of the unforgeability game (see checks of lines $5-8$ of the unforgeability experiment in Fig. 4.9), this index i^* must exist, $\mathcal{A}$ never makes a corruption query for i^*, and it does not ask for any $\mathsf{Sign}/\mathsf{Join}$ query involving i^* on message m_{j^*}. Whenever $\mathcal{B}$ discovers that it did not guess such indices correctly, $\mathcal{B}$ aborts. Nevertheless, since these indices can be kept perfectly hidden in $\mathcal{A}$'s view, $\mathcal{B}$ guesses these two indices with noticeable probability.

The next change consists into programming the random oracle to switch to an extraction-mode crs for the query on message m_{j^*}. Additionally, for each $j \neq j^*$, we can program the random oracle to output a $\mathsf{pk}_{\mathsf{O}_j}$ for which $\mathcal{B}$ knows the witness w_{1_j} s.t. $(x_{1_j}, w_{1_j}) \in \mathcal{R}_{\mathcal{L}}$. Every $\mathsf{Join}/\mathsf{Sign}$ query involving the signer i^* and a message m_j, with $j \neq j^*$, is answered using w_{1_j} instead of w_{i^*}. This change is not detectable by $\mathcal{A}$ thanks to the extended WI and the adaptive extractable soundness of ENIWI. Indeed, extended WI guarantees that $\mathcal{A}$ cannot notice the change of the used witness, and the adaptive extractable soundness guarantees that the probability of extracting a witness for statement x_{i^*} from the forgery does not change, except up to a negligible factor. Importantly, in order to reduce the indistinguishability of these changes to these two properties of the ENIWI we take advantage of the fact that we have a *different* CRS for every message. Finally, after applying all these changes, $\mathcal{B}$ can set x_{i^*} as the x received from $\mathcal{C}$. Given the forgery generated by $\mathcal{A}$, $\mathcal{B}$ can extract a witness for statement x, breaking the hardness of $\mathcal{R}_{\mathcal{L}}$.

Proof. We build an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to extract a witness for an instance of $\mathcal{R}_{\mathcal{L}}$ which is sampled by a challenger $\mathcal{C}$. In particular, $\mathcal{B}$ replies to all oracle queries of $\mathcal{A}$, including queries to the random oracle.

We prove unforgeability via a sequence of hybrid arguments. In each hybrid, $\mathcal{B}$ will change how it replies to certain queries, but in a way s.t. $\mathcal{A}$ cannot detect the change. In the final hybrid, $\mathcal{B}$ will be able to use a forgery from $\mathcal{A}$ to extract a valid witness for the instance sampled by $\mathcal{C}$.

$\mathcal{H}_0$: This is exactly the unforgeability game of figure Fig. 4.9.

$\mathcal{H}_1$: This is equivalent to $\mathcal{H}_0$ except that when replying to Join queries, $\mathcal{B}$ uses the Prove algorithm instead of the PrAdd algorithm. Additionally, before performing $\mathsf{AuxUpdate}$, each element of A is replaced with a fresh encryption of the auxiliary values in output of Prove algorithm. Let (k, Π_0, A_0) and (k, Π_1, A_1) the output of a Join query in $\mathcal{H}_0$ and $\mathcal{H}_1$ respectively. If $\mathcal{A}$ can distinguish between the two hybrids, $\mathcal{B}$ can use $\mathcal{A}$ to break the re-randomizable addition (Def. 30) of ENIWI. Let (Π', AUX') be the proof and auxiliary values (i.e., not encrypted) held by $\mathcal{B}$ before the Join query. Let $\mathcal{C}_{add}$ be the challenger of Def. 30. $\mathcal{B}$ sends $(x, w, \Pi', \mathsf{AUX}')$ ($\mathcal{B}$ can recover x, and w from $\mathsf{L_{keys}}$) to $\mathcal{C}_{add}$ and receives back (Π, AUX). Once $\mathcal{A}$ asks for a join operation, $\mathcal{B}$ sends (k, Π, A), where A is the encryption of the elements in AUX (again the encryption keys are available in $\mathsf{L_{keys}}$). This perfectly simulates the input of the adversary of the re-randomizable addition game (cfr., Def. 30).

$\mathcal{H}_2$: This is equivalent to $\mathcal{H}_1$ except that $\mathcal{B}$ tries to guess an index $j^* \in [q_m]$, where q_m is a bound on the number of random oracle queries. In particular, $\mathcal{B}$'s guess is that the j^*-th *different* queried message will be the one w.r.t. $\mathcal{A}$ outputs its forgery. If the message used by $\mathcal{A}$ in the forgery differs from m_{j^*}, then $\mathcal{B}$ aborts. The probability that $\mathcal{B}$ does not abort in this hybrid is at least $\frac{1}{q_m}$.

$\mathcal{H}_3$: This is equivalent to $\mathcal{H}_2$ except that $\mathcal{B}$ programs the random oracle on message m_{j^*} to give as output a CRS $\mathsf{crs_{Ext}}$ s.t. $(\mathsf{crs_{Ext}}, xk) \leftarrow \mathsf{Ext}_1(\mathsf{gk})$. $\mathcal{H}_3$ is indistinguishable from $\mathcal{H}_2$ due to the adaptive extractable soundness of ENIWI. (Def. 17). Indeed, $\mathsf{crs_{Ext}}$ is indistinguishable from a crs generated with $\mathsf{CRSSetup}$, which is in turn a random string.

$\mathcal{H}_4$ This is equivalent to $\mathcal{H}_3$ except that for every message m_j, with $j \neq j^*$, $\mathcal{B}$ programs the random oracle to give as output a random $\mathsf{pk}_{\mathsf{O}_j} = (x_{1_j}, \mathsf{pk}^1_{\mathsf{e_j}})$ for which $\mathcal{B}$ knows w_{1_j} s.t. $(x_{1_j}, w_{1_j}) \in \mathcal{R}_\mathcal{L}$. Since $\mathcal{R}_\mathcal{L}$ is public-coin samplable, $\mathsf{pk}_{\mathsf{O}_j}$ is equally distributed in $\mathcal{H}_3$ and $\mathcal{H}_4$.

$\mathcal{H}_5$: Let us consider the forgery given in output by $\mathcal{A}$ as $(k^*, m_{j^*}, \mathcal{R}^*, (k^*, \Pi^*, \mathsf{A}^*))$. Let $x_{\tilde{j^*}}$ be the trapdoor statement corresponding to message m_{j^*}[25] and $\{x_i\}_{i \in \mathcal{R}^*}$ be the statements corresponding to

[25]We use $\tilde{j^*}$ as a special index for the trapdoor statement related to m_{j^*}.

all the users in the ring of the forgery. For $\mathcal{A}$ to be admissible, there must be at least a statement $x_{i^*} \in \{x_i\}_{i \in \mathcal{R}^*} \cup \{x_{\tilde{j^*}}\}$ that was not involved in any *corrupt* query, or any $\mathsf{Join}/\mathsf{Sign}$ w.r.t. the forgery message m_{j^*}. If this does not hold, the checks of the unforgeability experiment of line 8 of Fig. 4.9 cannot be successful.

Let $\mathcal{H}_5$ be equivalent to $\mathcal{H}_4$ except that $\mathcal{B}$ tries to guess i^* sampling it uniformly at random from $\{i\}_{i \in [q_{KG}]} \cup \{\tilde{j^*}\}$, where q_{KG} is a polynomial bound on the number of key generation queries that $\mathcal{A}$ can do. When $\mathcal{A}$ outputs its forgery, $\mathcal{B}$ uses the extractor $(w_1, \ldots, w_{k^*}) \leftarrow \mathsf{Ext}_2(\mathsf{crs}_{\mathsf{Ext}}, xk, (k^*, \{x_i\}_{i \in \mathcal{R}^*} \cup \{x_{\tilde{j^*}}\}), \Pi^*)$ to extract the witnesses from proof Π^*. If the extraction fails, or none of the extracted witnesses w_z, with $z \in [k^*]$, is s.t. $(x_{i^*}, w_z) \in \mathcal{R}_{\mathcal{L}}$, $\mathcal{B}$ aborts. Due to the adaptive extractable soundness of ENIWI, the extraction fails only with negligible probability. Thus, the probability that $\mathcal{B}$ does not abort in this hybrid is at least $\frac{1}{q_{KG}+1}$.

$\mathcal{H}_6$: This is equivalent to $\mathcal{H}_5$ except that every time $\mathcal{A}$ makes Sign or Join queries involving i^* for message $m_j \neq m_{j^*}$, $\mathcal{B}$ answers using the witness w_1^j for the trapdoor statement x_{1_j} to compute the proof and the auxiliary values while still encrypting $\perp$ in a_{i^*}. The queries for message $m_j = m_{j^*}$ are answered in the same way, since no query for message m_{j^*} ever involves i^*.

If i^* is equal to $\tilde{j^*}$, then $\mathcal{H}_5$ and $\mathcal{H}_6$ are equally distributed since no Sign or Join query can involve $x_{\tilde{j^*}}$ by construction. Let us consider the case for which i^* is related to a registered key. We now argue that $\mathcal{H}_5$ is indistinguishable from $\mathcal{H}_6$ thanks to extended WI property (Def. 32). Let us call $\mathcal{D}$ the distinguisher that distinguishes $\mathcal{H}_5$ from $\mathcal{H}_6$ with probability greater than negligible. Let $\mathcal{A}_{\mathsf{EWI}}$ be the adversary that exploits $\mathcal{D}$ to break the extended WI of ENIWI against a challenger $\mathcal{C}_{\mathsf{EWI}}$. W.l.o.g. we consider a query $\mathsf{OSign}(m_j, \mathcal{R}, i^*)$ for message m_j, with $j \neq j^*$, and signer index i^*. Trivially, it can be extended to any query involving i^*.

1. $\mathcal{A}_{\mathsf{EWI}}$ chooses as statement x the public keys of the ring $\mathcal{R}$,

$\mathcal{A}_{\mathsf{EWI}}$ chooses $w_0 = w_{1_{\tilde{j}}}$ and $w_1 = w_{i^*}$.

2. $\mathcal{A}_{\mathsf{EWI}}$ sends (x, w_0, w_1) to the challenger of extended WI $\mathcal{C}_{\mathsf{EWI}}$, and receives back (Π, AUX)[26].
3. $\mathcal{A}_{\mathsf{EWI}}$ generates the signature starting from (Π, AUX) as in the Sign algorithm (except for the fact that a_{i^*} is obtained encrypting $\perp$) and sends this signature as answer to the query $\mathsf{OSign}(m_j, \mathcal{R}, i^*)$ performed by $\mathcal{A}$.
4. At the end of the experiment, $\mathcal{A}$ outputs the forgery $(k^*, m_{j^*}, \mathcal{R}^*, (k^*, \Pi^*, \mathsf{A}^*))$ to $\mathcal{A}_{\mathsf{EWI}}$.
5. $\mathcal{A}_{\mathsf{EWI}}$ runs $\mathcal{D}$ on input its view.$\mathcal{A}_{\mathsf{EWI}}$ returns 1 if $\mathcal{D}$ says that the hybrid is $\mathcal{H}_5$, and 0 otherwise.

$\mathcal{A}_{\mathsf{EWI}}$ breaks the extended WI property with the same advantage that $\mathcal{D}$ has in distinguishing the two hybrids. Therefore, $\mathcal{H}_5$ and $\mathcal{H}_6$ are indistinguishable[27]. Additionally, we now argue that the probability of extracting a witness for i^* does not change between the two hybrids, except for a negligible quantity. To see why this holds, let us consider an $\mathcal{A}_{\mathsf{EWI}}$ that at step 5., runs the extractor $\mathsf{Ext}_2(\mathsf{crs}_{\mathsf{Ext}}, xk, (k^*, \{x_i\}_{i \in \mathcal{R}^*} \cup \{x_{\tilde{j}^*}\}), \Pi^*)$ to extract the witnesses $(w_1, \ldots, w_{k^*})$ used in the forgery instead of calling $\mathcal{D}$. If the forgery contains w_{i^*} s.t. $(x_{i^*}, w_{i^*}) \in \mathcal{R}_{\mathcal{L}}$, $\mathcal{A}_{\mathsf{EWI}}$ returns 1, and 0 otherwise. If $|\Pr[\mathcal{A}_{\mathsf{EWI}}$ outputs $1|\mathcal{H}_5] - \Pr[\mathcal{A}_{\mathsf{EWI}}$ outputs $1|\mathcal{H}_6]| > \mathsf{negl}(\lambda)$, then $\mathcal{A}_{\mathsf{EWI}}$ would itself be a distinguisher between the two hybrids. Therefore, since this would break the extended WI property, we reach a contradiction. Importantly, we exploit the fact that we have a *different* CRS for every message. Indeed, we only switch one CRS to the extraction mode; namely the one corresponding to the forgery message m_{j^*}. This allows us to simultaneously reduce to EWI

[26]Notice that here AUX does not include aux_{i^*} and $\mathsf{aux}_{\tilde{j}}$.

[27]Notice that this reduction in principle does not capture the fact that $\mathcal{A}$ can perform multiple sequential queries in a single execution. Consider a modified EWI definition with a game in which the challenger accepts multiple queries and always replies according to the bit sampled at the beginning of the game. It is straightforward to observe that a proof system which fulfils the regular EWI definition also fulfils the modified EWI definition.

for all the queries on messages $m_j \neq m_{j^*}$, and to extract the witnesses from the forgery on message m_{j^*} in order to finalize the game as shown in the next hybrid.

$\mathcal{H}_7$: This is equivalent to $\mathcal{H}_6$ except that when prompted to generate a key pair for i^*, $\mathcal{B}$ would return as public key the pair $(x, \mathsf{pk_e})$, where x, instead of being freshly sampled, is the one that $\mathcal{C}$ sent to $\mathcal{B}$ in the hard relation game. $\mathcal{H}_6$ and $\mathcal{H}_7$ are equally distributed. Now, $\mathcal{B}$ extracts from the forgery a witness for the hard relation sampled by the challenger of the hard relation game, thus reaching a contradiction.

□

Lemma 7. *The signature scheme described in Fig. 4.13 and Fig. 4.14 satisfies the anonymity property of Def. 26.*

Proof sketch. Through a sequence of indistinguishable hybrids, we switch from a challenger $\mathcal{B}$ using lad_0 to a $\mathcal{B}$ using lad_1. We show that at every hybrid, $\mathcal{B}$ can exploit $\mathcal{A}$ distinguishing between the two hybrids to break some properties of the underlying primitives. First, $\mathcal{B}$ changes the way it processes the ladders and replies to Join queries. In particular, $\mathcal{B}$ computes every time a new proof from scratch using $\mathsf{ENIWI.Prove}$, instead of running the $\mathsf{Join}/\mathsf{Extend}$ algorithms, analogously to the proof of unforgeability. After that, when processing the ladders, $\mathcal{B}$ will encrypt $\perp$ in all signers' ciphertexts. This change is not detected by $\mathcal{A}$ thanks to the IND-CPA property of the encryption scheme. At the end, $\mathcal{B}$ fixes the ladder used in the anonymity game to be lad_1. This change is unnoticeable thanks to the extended WI of ENIWI.

Proof. **$\mathcal{H}_0$:** This is exactly the anonymous extendability game of Fig. 4.10.

$\mathcal{H}_1$: This is equivalent to $\mathcal{H}_0$ except that when running the Join algorithm, $\mathcal{B}$ uses the Prove algorithm instead of the PrAdd algorithm. Additionally, instead of performing $\mathsf{AuxUpdate}$, each element of A is replaced with a fresh encryption of the auxiliary values in output of Prove algorithm. Showing that $\mathcal{H}_1$ is indistinguishable from $\mathcal{H}_0$ basically mirrors the discussion of $\mathcal{H}_1$ in Lem. 6.

$\mathcal{H}_2$: This is equivalent to $\mathcal{H}_1$ except that when running the Extend algorithm in the ladder, $\mathcal{B}$ uses the Prove algorithm instead of the PrExtend algorithm. Additionally, instead of performing AuxUpdate, each element of A is replaced with a fresh encryption of the auxiliary values in output of Prove algorithm. Showing that $\mathcal{H}_2$ is indistinguishable from $\mathcal{H}_1$ basically mirrors the discussion for the previous hybrid, except that now we would use $\mathcal{A}$ to break the re-randomizable extension property (cfr., Def. 31) of ENIWI.

$\mathcal{H}_3$: This is equivalent to $\mathcal{H}_2$ except that, when processing the ladders, $\mathcal{B}$ encrypts $\perp$ in all the signers' ciphertexts. Note that auxiliary values are already not used at all by $\mathcal{B}$ at this point. Recall that in order to win, $\mathcal{A}$ cannot corrupt any of the signers. This is indistinguishable from $\mathcal{H}_2$ thanks to the IND-CPA property of the encryption scheme [28].

$\mathcal{H}_4$: This is equivalent to $\mathcal{H}_3$ except that $\mathcal{B}$ does not sample b at random but fixes $b = 0$. If $b = 0$ also in $\mathcal{H}_3$, the hybrids are identical. Let us assume that $b = 1$ in $\mathcal{H}_3$. Let $\Sigma^{\mathcal{H}_j}$ be the challenge list given to $\mathcal{A}$ at the end of $\mathcal{H}_j$, with $j \in \{3, 4\}$. Let us denote the i-th element of $\Sigma^{\mathcal{H}_j}$ as $\sigma_i^{\mathcal{H}_j} = (k, \Pi_i^{\mathcal{H}_j}, \mathsf{A}_i^{\mathcal{H}_j})$, where $i \in [l]$ and l is the length of the ladders. We now argue that $\mathcal{A}$ cannot distinguish $\mathcal{H}_4$ from $\mathcal{H}_3$. First, since the ladders are well-formed all the $\sigma_i^{\mathcal{H}_3}$ and $\sigma_i^{\mathcal{H}_4}$ contain an ENIWI w.r.t. the same statement. Let us assume $\mathcal{A}$ corrupted all the keys that are not in

[28] Let us consider an extended version of the IND-CPA game that is trivially implied by the standard IND-CPA game. In this extended game, the adversary $\mathcal{A}$ has access to a key generation and a corruption oracle. Let n be the number of key generation queries. At the end of this step, $\mathcal{A}$ outputs a pair of messages (m_0, m_1) and an index i. $\mathcal{C}$ checks that i is not corrupted, and samples a random bit $b \leftarrow\!\!\$ \{0, 1\}$. $\mathcal{C}$ encrypts m_b using public key pk_e^i, $\mathcal{A}$ wins if it guesses the value of b. Indistinguishability of $\mathcal{H}_3$ and $\mathcal{H}_2$ can be proven via a reduction to this extended IND-CPA game through a sequence of hybrids in which one ciphertext at a time is modified to encrypt $\perp$.

the ladders[29]. Therefore, $\mathcal{A}$ gets to see the corresponding auxiliary values. As a result, an $\mathcal{A}$ distinguishing between $\mathcal{H}_4$ and $\mathcal{H}_3$ would get access to two ENIWI with auxiliary values associated to keys whose corresponding witnesses were not used in any of the two ENIWI. In addition to that, $\mathcal{A}$ just gets encryptions of values that are uncorrelated with the two ENIWI and the auxiliary values. It follows that we can directly reduce the indistinguishability between $\mathcal{H}_4$ and $\mathcal{H}_3$ to the security game of extended witness indistinguishability (cfr., Def. 32).

□

Lemma 8. *The signature scheme described in Fig. 4.13 and Fig. 4.14 enjoys fellow signer anonymity (cfr., Def. 27).*

The proof follows essentially the same path of the one of Lem. 7.

4.2.10 Our Extendable Non-Interactive Witness Indistinguishable Proof of Knowledge

In this section, we first show how to extend the GS proof system to define a proof system for a threshold relation. After that, we show how to further modify such scheme to get our ENIWI PoK.

4.2.11 GS Proofs of Partial Satisfiability

In [31, 69], it is shown how to transform n sets of certain types of equations $S_1, \ldots, S_n$ to a set of equations S' s.t. S' is satisfied whenever one of $S_1, \ldots, S_n$ is satisfied. A witness for S_i, with $i \in [n]$, is easily mapped to a witness for S'. Indeed, this transformation realizes a disjunction. The transformation works by assuming that $S_1, \ldots, S_n$ have independent variables, adding variables $b_1, \ldots b_{n-1} \in \{0, 1\}$, and defining $b_n = 1 - b_1 - \ldots - b_{n-1}$. Then, for $i \in [n]$, b_i is used to modify all the equations in S_i so that they remain the same if $b_i = 1$, but they admit the trivial solution for $b_i = 0$. Slightly increasing the overhead of these compilers, it is also possible to implement partial satisfiability proofs for an arbitrary threshold k, meaning that S' is

[29]These are the only keys $\mathcal{A}$ can corrupt while still being admissible.

satisfied iff k of $S_1, \ldots, S_n$ are satisfied. To do so, the main idea is to define $b_n \in \{0, 1\}$, and to prove that $b_1 + \ldots + b_n = k$.

A case which is relevant to this paper is when $S_1, \ldots, S_n$ contain only PPEs with $t_{\mathbb{T}} = 0_{\mathbb{T}}$, all the variables of the PPEs are elements of $\check{\mathbb{H}}$, and public constants are either paired with secret values or with $\check{h}$. In this case, the prover would:

1. Add variables $b_1, \ldots, b_n$ and prove that $b_i \in \{0, 1\}\ \forall i \in [n]$. This can be done with quadratic equations, by adding the equations $b_i(1 - b_i) = 0$. Let us define such equations to be of type $\mathcal{B}$, we will refer to a specific equation using $\mathcal{B}_i$.

2. Add variables $\hat{M}_1, \ldots, \hat{M}_n$ and prove $b_i\hat{g} - \hat{M}_i = 0$, with $i \in [n]$. This can be done via multi-scalar multiplication equations in $\hat{\mathbb{G}}$. Since $b_i \in \{0, 1\}$, it follows that $\hat{M}_i \in \{\hat{0}, \hat{g}\}$. Let us define such equations to be of type $\mathcal{M}$.

3. Add equation $\sum_{i=1}^{n} \hat{M}_i \cdot \check{h} - k\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$. Since $\hat{M}_i \in \{\hat{0}, \hat{g}\}$, this equation implies that exactly k of the $\hat{M}_i$, with $i \in [n]$, are equal to $\hat{g}$. Let us call such equation as $\mathcal{K}$.

4. For each S_i, with $i \in [n]$, let Q_i be the number of equations in S_i, let $J_{i,q}$ be the number of variables in the equation $q \in [Q_i]$ of S_i. For each variable $\check{y}_{i,q,j}$ with $q \in [Q_i], j \in [J_{i,q}]$, define variable $\check{x}_{i,q,j}$ and add equation $\hat{M}_i \cdot \check{y}_{i,q,j} - \hat{M}_i \cdot \check{x}_{i,q,j} = 0_{\mathbb{T}}$. Since k of the $\hat{M}_i$ are equal to $\hat{g}$, this implies that for k equations sets it must hold that all $\check{y}_{i,q,j} = \check{x}_{i,q,j}$. Let us define such equations to be of type $\mathcal{Y}$.

5. For each equation in each S_i, replace all the original $\check{y}_{i,q,j}$ with the corresponding $\check{x}_{i,q,j}$. This allows to set all $\check{x}_{i,q,j} = \check{y}_{i,q,j} = \check{0}$ for each set S_i for which the prover does not have a satisfying assignment. For the k sets for which the prover does have a satisfying assignment, the prover sets $\check{y}_{i,q,j} = \check{x}_{i,q,j}$. Let us define such equations to be of type $\mathcal{X}$.

4.2.12 High-level Overview of our ENIWI.

We construct our ENIWI by observing that GS proofs of partial satisfiability can be updated in two ways:

- **Extend**: consider a proof Π for a set of equations S which is satisfied if k out of n of the original equations sets $S_1, \ldots, S_n$ are satisfied. On input a new equations set S_{n+1} and the proof Π, compute a new equations set S' which is satisfied if k out of the $n+1$ equations sets $S_1, \ldots, S_n, S_{n+1}$ are satisfied. Output S' and the corresponding updated proof Π'.

- **Add**: consider a proof Π for a set of equations S which is satisfied if k out n of the original equations sets $S_1, \ldots, S_n$ are satisfied. On input the proof Π for S, a witness for an equations set S_i with $i \in [n]$ which was not previously used to create Π, and some corresponding auxiliary information aux_i, compute a new equations set S' which is satisfied if $k+1$ out of the n equations sets $S_1, \ldots, S_n$ are satisfied. Output S' and the corresponding updated proof Π'.

In particular, one can notice that each step of the partial satisfiability proof described in Sec. 4.2.11 only adds equations featuring independent variables, except for step 3. In step 3, one equation is added combining all variables $\hat{M}_i$ with $i \in [n]$. The equation is $\sum_{i=1}^{n} \hat{M}_i \cdot \check{h} - k\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$. Let us compute the GS proof for such equation. Let crs be $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \check{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}})$.

- Variables $\hat{M}_i$ are committed as group elements (i.e., with label $\mathsf{com}_{\hat{\mathbb{G}}}$), thus $\hat{c}_{\hat{M}_i} = \boldsymbol{e}^\top \hat{M}_i + \hat{\boldsymbol{v}} r_i + \hat{\boldsymbol{w}} s_i$, with $r_i, s_i \leftarrow\$ \mathbb{Z}_p$.

- $\hat{g}$ is the base element of $\hat{\mathbb{G}}$, thus it is publicly committed with label $\mathsf{base}_{\hat{\mathbb{G}}}$ as $\hat{c}_{\hat{g}} = (0, \hat{g})^\top$.

- $\check{h}$ is the base element of $\check{\mathbb{H}}$, and thus it is publicly committed with label $\mathsf{base}_{\check{\mathbb{H}}}$ as $(0, \check{h})$.

This results in $\hat{C} = (\hat{c}_{\hat{M}_1}, \ldots, \hat{c}_{\hat{M}_n}, \hat{c}_{\hat{g}})$, $\check{D} = (0, \check{h})$, $\boldsymbol{r}_x = (r_1, \ldots, r_n, 0)^\top$, $\boldsymbol{s}_x = (s_1, \ldots, s_n, 0)^\top$, $\boldsymbol{r}_y = 0$, $\boldsymbol{s}_y = 0$.

This means that $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = -\hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = -\hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$, with $\alpha, \gamma, \beta, \delta$ being random elements in $\mathbb{Z}_p$.

Let us compute $\boldsymbol{r}_x \Gamma \check{D} = (r_1, \ldots, r_n, 0)^\top (1, \ldots, 1, -k)(0, \check{h}) = (0, \sum_{i=1}^n r_i \check{h})$. Similarly, we have that $\boldsymbol{s}_x \Gamma \check{D} = (0, \sum_{i=1}^n s_i \check{h})$. Let us define $\mathsf{aux}_i = (\mathsf{aux}_i^1, \mathsf{aux}_i^2) = (r_i \check{h}, s_i \check{h})$. This means that $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \boldsymbol{r}_x \Gamma \check{D} + \alpha \check{\boldsymbol{v}} + \beta \check{\boldsymbol{w}} = (0, \sum_{i=1}^n \mathsf{aux}_i^1) + \alpha \check{\boldsymbol{v}} + \beta \check{\boldsymbol{w}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \boldsymbol{s}_x \Gamma \check{D} + \delta \check{\boldsymbol{v}} + \gamma \check{\boldsymbol{w}} = (0, \sum_{i=1}^n \mathsf{aux}_i^2) + \delta \check{\boldsymbol{v}} + \gamma \check{\boldsymbol{w}}$.

We notice that the proof elements for equation $\mathcal{K}$ are essentially a sum of n independent contributions (i.e., the aux_i values) for each of the involved n variables (i.e., $\hat{M}_i$ with $i \in [n]$). We can exploit this fact to perform the extend and add operations in the following way. Let us consider the steps of Sec. 4.2.11.

- **Extend**: Add new equations of types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$ by defining the corresponding new independent variables, and compute the related GS proofs. Modify equation $\mathcal{K}$ to be $\sum_{i=1}^{n+1} \hat{M}_i \cdot \check{h} - k\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$ and update $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}$ as $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} + (0, r_{n+1}\check{h}), \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} + (0, s_{n+1}\check{h})$, where r_{n+1} and s_{n+1} are the randomnesses used to commit to the new variable $\hat{M}_{n+1} = \hat{0}$.

- **Add**: Replace the committed variables for the equations $\mathcal{B}_i, \mathcal{M}_i, \mathcal{Y}_i, \mathcal{X}_i$ with new committed variables $b_i = 1$, $\hat{M}_i = \hat{g}$, and $\check{y}_{i,q,j} = \check{x}_{i,q,j}$. Replace the old corresponding GS proofs with freshly computed ones. Modify equation $\mathcal{K}$ to be $\sum_{i=1}^n \hat{M}_i \cdot \check{h} - (k+1)\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$, and update $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}$ as $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} - (0, \mathsf{aux}_i^1) + (0, r_i'\check{h}), \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} - (0, \mathsf{aux}_i^2) + (0, s_i'\check{h})$, where r_i' and s_i' are the randomnesses used for the fresh commitment to $\hat{M}_i = \hat{g}$.

It is pretty straightforward to notice that, after any of the two above modifications, the resulting proof is an accepting proof for the updated threshold relation. Indeed, both the extend and add operation symbolically compute the proofs in the same way a prover for the updated threshold relation would do from scratch.

4.2.13 Our ENIWI

We now describe our ENIWI in detail. In particular, it is an ENIWI PoK over the language of sets of pairing product equations where all the variables are elements of $\check{\mathbb{H}}$, public constants are either paired with secret values or with $\check{h}$, and the target element is $0_{\mathbb{T}}$. For simplicity, we consider each statement x_i as containing only one equation.

- $\mathsf{crs} \leftarrow \mathsf{CRSSetup}(\mathsf{gk})$: run $\mathsf{GS.Setup}(\mathsf{gk})$. This results in $\mathsf{crs} = (\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \check{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}})$.

- $(\Pi, (\mathsf{aux}_1, \ldots, \mathsf{aux}_n)) \leftarrow \mathsf{Prove}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)))$: on input $((k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k))) \in \mathcal{R}_t$, define $A = \{\alpha_1, \ldots, \alpha_k\}$[30] and do the following.

  1. For each equation x_i, $i \in [n]$, define new variables and equations:
     - Define variable $b_i = 1$ if $i \in A$, and $b_i = 0$ otherwise.
     - Define quadratic equation $\mathcal{B}_i$ as $b_i(1 - b_i) = 0$.
     - Define variables $\hat{M}_i = \hat{g}$ if $i \in A$, and $\hat{M}_i = \hat{0}$ otherwise.
     - Define multi-scalar multiplication equation $\mathcal{M}_i$ as $b_i\hat{g} - \hat{M}_i = 0$.
     - Let J_i be the number of variables in equation x_i. For each variable $\check{y}_{i,j}$, with $j \in [J_i]$, define a variable $\check{x}_{i,j}$. Set $\check{x}_{i,j} = \check{y}_{i,j}$, if $i \in A$, and $\check{x}_{i,j} = \check{0}$ otherwise.
     - For each variable $\check{y}_{i,j}$, with $j \in [J_i]$, define pairing product equation $\mathcal{Y}_{i,j}$ as $\hat{M}_i \cdot \check{y}_{i,j} - \hat{M}_i \cdot \check{x}_{i,j} = 0_{\mathbb{T}}$.
     - Modify pairing product equation x_i by replacing each variable $\check{y}_{i,j}$, with $j \in [J_i]$, with variable $\check{x}_{i,j}$. Let us call such modified equation $\mathcal{X}_i$.

     Moreover, define pairing product equation $\mathcal{K}$ as $\sum_{i=1}^{n} \hat{M}_i \cdot \check{h} - k\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$. At the end of this step, there will be n equations of types $\mathcal{B}, \mathcal{M}, \mathcal{X}$, $n \sum_{i=1}^{n} J_i$ equations of type $\mathcal{Y}$, and one equation of type $\mathcal{K}$.

  2. For each equation of types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$ generate appropriate commitments (using $\mathtt{GS.com}$) to all variables, resulting

[30] A indicates what are the k equations the prover has a satisfying assignment for.

in lists of commitments $\boldsymbol{C_{\mathcal{B}}}, \boldsymbol{C_{\mathcal{M}}}, \boldsymbol{C_{\mathcal{Y}}}, \boldsymbol{C_{\mathcal{X}}}$ respectively[31]. Then, for each equation of types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$, run $\mathsf{GS.Prove}$ with the obvious inputs obtaining proof elements lists $\boldsymbol{\pi_{\mathcal{B}}}$, $\boldsymbol{\pi_{\mathcal{M}}}, \boldsymbol{\pi_{\mathcal{Y}}}, \boldsymbol{\pi_{\mathcal{X}}}$. For example, $\boldsymbol{\pi_{\mathcal{B}}}$ contains proof elements $\boldsymbol{\pi_{\mathcal{B}}}_i$, with $i \in [n]$, each of them obtained running $\mathsf{GS.Prove}$ for equation $\mathcal{B}_i$ using commitments $\boldsymbol{C_{\mathcal{B}}}_i$ (and related randomnesses) from $\boldsymbol{C_{\mathcal{B}}}$.

Moreover, for equation $\mathcal{K}$ do the following[32]:

- Commit to $\hat{M}_i$, with $i \in [n]$, with label $\mathsf{com}_{\hat{\mathbb{G}}}$ and randomness (r_i, s_i), i.e., $(\mathsf{com}_{\hat{\mathbb{G}}}, \hat{c}_{\hat{M}_i}) \leftarrow \mathsf{GS.com}(\mathsf{com}_{\hat{\mathbb{G}}}, \hat{M}_i; (r_i, s_i))$, resulting in $\hat{c}_{\hat{M}_i} = \boldsymbol{e}^\top \hat{M}_i + \hat{\boldsymbol{v}} r_i + \hat{\boldsymbol{w}} s_i$.
- Commit to $\hat{g}$ with label $\mathsf{base}_{\hat{\mathbb{G}}}$ and randomness $(0, 0)$, i.e., $(\mathsf{base}_{\hat{\mathbb{G}}}, \hat{c}_{\hat{g}}) \leftarrow \mathsf{GS.com}(\mathsf{base}_{\hat{\mathbb{G}}}, \hat{g}; (0, 0))$, resulting in $\hat{c}_{\hat{g}} = (0, \hat{g})^\top$.
- Commit to $\check{h}$ with label $\mathsf{base}_{\check{\mathbb{H}}}$ and randomness $(0, 0)$, i.e., $(\mathsf{base}_{\check{\mathbb{H}}}, \check{d}_{\check{h}}) \leftarrow \mathsf{GS.com}(\mathsf{base}_{\check{\mathbb{H}}}, \check{h}; (0, 0))$, resulting in $\check{d}_{\check{h}} = (0, \check{h})$.

Do the following steps:

- Define $\hat{C} = (\hat{c}_{\hat{M}_1}, \ldots, \hat{c}_{\hat{M}_n}, \hat{c}_{\hat{g}}), \check{D} = (0, \check{h}), \boldsymbol{r}_x = (r_1, \ldots, r_n, 0)^\top, \boldsymbol{s}_x = (s_1, \ldots, s_n, 0)^\top, \boldsymbol{r}_y = 0, \boldsymbol{s}_y = 0$. This means that $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}} = -\hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$ and $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}} = -\hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$.
- Compute $\boldsymbol{r}_x \Gamma \check{D} = (r_1, \ldots, r_n, 0)^\top (1, \ldots, 1, -k)(0, \check{h}) = (0, \sum_{i=1}^n r_i \check{h})$. Similarly, we have that $\boldsymbol{s}_x \Gamma \check{D} = (0, \sum_{i=1}^n s_i \check{h})$. Define $\mathsf{aux}_i = (\mathsf{aux}_i^1, \mathsf{aux}_i^2) = (r_i \check{h}, s_i \check{h})$, with $i \in [n]$.
- Compute $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \boldsymbol{r}_x \Gamma \check{D} + \alpha \check{\boldsymbol{v}} + \beta \check{\boldsymbol{w}} = (0, \sum_{i=1}^n \mathsf{aux}_i^1) + \alpha \check{\boldsymbol{v}} + \beta \check{\boldsymbol{w}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \boldsymbol{s}_x \Gamma \check{D} + \delta \check{\boldsymbol{v}} + \gamma \check{\boldsymbol{w}} = (0, \sum_{i=1}^n \mathsf{aux}_i^2) + \delta \check{\boldsymbol{v}} + \gamma \check{\boldsymbol{w}}$.

[31] Whenever different equations share the same variables, we can think of the commitments lists as containing copies of the exact same commitments. Clearly, in practice data does not need to be replicated.

[32] We report the whitebox computation of the GS prover to show how to compute the auxiliary values. Furthermore, for sake of clarity, we report again commitments to variables $\hat{M}_i$ with $i \in [n]$, which were already created to prove other equations.

Let $\pi_{\mathcal{K}} = (\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}})$ and $C_{\mathcal{K}} = (\hat{C}, \check{D})$. Output $(\Pi = (\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}, \pi_{\mathcal{K}}), \mathsf{AUX} = (\mathsf{aux}_1, \ldots, \mathsf{aux}_n))$.

- $0/1 \leftarrow \mathsf{PrVerify}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi)$ reconstruct equations of type $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}, \mathcal{K}$, appropriately parse Π, and for every equation run $\mathsf{GS.PrVerify}$ with the obvious inputs. For example, the proof for equation $\mathcal{B}_i$ is verified giving, after appropriate parsing, commitments $\boldsymbol{C}_{\boldsymbol{\mathcal{B}}i}$ and proof element $\boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}i}$ in input to $\mathsf{GS.PrVerify}$. Return 1 iff all the calls to $\mathsf{GS.PrVerify}$ return 1.

- $(\Pi', \mathsf{aux}_{n+1}) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, (k, x_1, \ldots, x_n), x_{n+1}, \Pi)$ do the following:

  1. Parse Π as $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}, \pi_{\mathcal{K}})$, $\mathsf{AUX} = (\mathsf{aux}_1, \ldots, \mathsf{aux}_n)$.
  2. For each of the 4 equation types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$, add a new equation related to x_{n+1} by defining the corresponding new independent variables, $b_{n+1} = 0, \hat{M}_{n+1} = \hat{0}$ and all the $\check{y}_{n+1,j} = \check{0}$, with $j \in [J_{n+1}]$.
  3. Compute commitments to new variables and appropriately add them to $\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}$.
  4. Compute the related new GS proofs and add them to $\boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}$, $\boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}$ accordingly.
  5. Parse $\pi_{\mathcal{K}}$ as $(\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}})$ and update $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}$ as $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} + (0, r_{n+1}\check{h}), \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} + (0, s_{n+1}\check{h})$, where r_{n+1} and s_{n+1} are the randomnesses used to commit to the new variable $\hat{M}_{n+1} = \hat{0}$.
  6. Set $\mathsf{aux}_{n+1} = (\mathsf{aux}^1_{n+1}, \mathsf{aux}^2_{n+1}) = (r_{n+1}\check{h}, s_{n+1}\check{h})$.
  7. Output $(\Pi, \mathsf{aux}_{n+1})$.

- $(\Pi', \mathsf{aux}'_{\alpha}) \leftarrow \mathsf{PrAdd}(\mathsf{crs}, (k, x_1, \ldots, x_n), (w, \alpha), \mathsf{aux}, \Pi)$ do the following:

  1. Parse Π as $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}, \pi_{\mathcal{K}})$.

2. For each of the 4 equation types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$, replace the variables in equations related to x_α (i.e, $\mathcal{B}_\alpha, \mathcal{M}_\alpha, \mathcal{X}_\alpha$, and all $\mathcal{Y}_{\alpha,j}$ with $j \in J_\alpha$) as follows: $b_\alpha = 1, \hat{M}_\alpha = \hat{g}$ and all the $\check{y}_{\alpha,j} = \check{x}_{\alpha,j}$, with $j \in [J_\alpha]$.
3. Replace the commitments related to equations $\mathcal{B}_\alpha, \mathcal{M}_\alpha, \mathcal{X}_\alpha$, and all $\mathcal{Y}_{\alpha,j}$, with $j \in J_\alpha$ with freshly generated ones updating $\boldsymbol{C_{\mathcal{B}}}, \boldsymbol{C_{\mathcal{M}}}, \boldsymbol{C_{\mathcal{Y}}}, \boldsymbol{C_{\mathcal{X}}}$ accordingly.
4. Replace the GS proofs related to equations $\mathcal{B}_\alpha, \mathcal{M}_\alpha, \mathcal{X}_\alpha$, and all $\mathcal{Y}_{\alpha,j}$ with $j \in J_\alpha$, with freshly generated ones replacing proof elements of $\boldsymbol{\pi_{\mathcal{B}}}, \boldsymbol{\pi_{\mathcal{M}}}, \boldsymbol{\pi_{\mathcal{Y}}}, \boldsymbol{\pi_{\mathcal{X}}}$ accordingly.
5. Parse $\pi_{\mathcal{K}}$ as $(\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}})$ and update $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}$ as $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} - (0, \mathsf{aux}^1_\alpha) + (0, r'_\alpha \check{h}), \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} - (0, \mathsf{aux}^2_\alpha) + (0, s'_\alpha \check{h})$, where r'_α and s'_α are the randomnesses used for the fresh commitment to $\hat{M}_\alpha = \hat{g}$.
6. Set $\mathsf{aux}'_\alpha = (\mathsf{aux}^1_\alpha, \mathsf{aux}^2_\alpha) = (r'_\alpha \check{h}, s'_\alpha \check{h})$.
7. Output $(\Pi, \mathsf{aux}'_\alpha)$.

- $(\Pi', r_1, \ldots, r_n) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi)$:

  1. Run $\mathsf{GS.RandPr}$ on each of the proofs, appropriately fixing the random coins when randomizing proofs related to equations involving shared variables (i.e., s.t. we end up again with shared variables having the exact same commitments). Let r'_i, s'_i, with $i \in [n]$ be the randomnesses used to update commitments to all $\hat{M}_i$, with $i \in [n]$. Define $r_i = (r'_i, s'_i)$. Let randomized proof elements and commitments be contained in Π'.
  2. Output $(\Pi', r_1, \ldots, r_n)$

- $\mathsf{aux}' \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{aux}, r)$:

  1. Parse r as (r', s'), and aux as $(\mathsf{aux}^1, \mathsf{aux}^2)$.
  2. Output $\mathsf{aux}' = (\mathsf{aux}^1 + r'\check{h}, \mathsf{aux}^2 + s'\check{h})$.

- $0/1 \leftarrow \mathsf{AuxVerify}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)), (\mathsf{aux}_1, \ldots, \mathsf{aux}_n), \Pi)$:

1. Parse Π as $(\boldsymbol{C_{\mathcal{B}}}, \boldsymbol{C_{\mathcal{M}}}, \boldsymbol{C_{\mathcal{Y}}}, \boldsymbol{C_{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi_{\mathcal{B}}}, \boldsymbol{\pi_{\mathcal{M}}}, \boldsymbol{\pi_{\mathcal{Y}}}, \boldsymbol{\pi_{\mathcal{X}}}, \pi_{\mathcal{K}})$. Parse $C_{\mathcal{K}}$ as $\hat{C} = (\hat{c}_{\hat{M}_1}, \ldots, \hat{c}_{\hat{M}_n}, \hat{c}_{\hat{g}})$ and $\check{D} = (0, \check{h})$.
2. Check that $(\mathsf{aux}_{\alpha_1}, \ldots, \mathsf{aux}_{\alpha_k})$ all open $(\hat{c}_{\hat{M}_{\alpha_1}}, \ldots, \hat{c}_{\hat{M}_{\alpha_k}})$ to $\hat{g}$. Namely, check that $\hat{c}_{\hat{M}_i} \cdot (\check{h}, \check{h}) + \hat{\boldsymbol{v}} \cdot (-\mathsf{aux}_i^1, -\mathsf{aux}_i^1) + \hat{\boldsymbol{w}} \cdot (-\mathsf{aux}_i^2, -\mathsf{aux}_i^2) = (\hat{0}, \hat{g})^\top \cdot (\check{h}, \check{h})$, for all $i \in A$.
3. Check that remaining auxiliary values open commitments $\hat{c}_{\hat{M}_i}$ with $i \in [n] \setminus A$ to $\hat{0}$. Namely, check that $\hat{c}_{\hat{M}_i} \cdot (\check{h}, \check{h}) + \hat{\boldsymbol{v}} \cdot (-\mathsf{aux}_i^1, -\mathsf{aux}_i^1) + \hat{\boldsymbol{w}} \cdot (-\mathsf{aux}_i^2, -\mathsf{aux}_i^2) = (\hat{0}, \hat{0})^\top \cdot (\check{h}, \check{h})$, for all $i \in [n] \setminus A$.

Theorem 3. *If* GS *(cfr., Sec. 4.2.5) is a NIWI for all equation types and a NIWI PoK for pairing product equations, then the construction above is an* ENIWI *PoK. The base relation $\mathcal{R}_{\mathcal{L}}$ consists of pairing product equations in which all the variables are elements of $\check{\mathbb{H}}$, public constants are either paired with secret values or with $\check{h}$, and the target element is $0_{\mathbb{T}}$.*

Lemma 9. *The proof system described above enjoys completeness and transformation completeness.*

Proof. Due to the completeness of GS, for honestly generated proofs the output of PrVerify is always 1. Let us prove that the same holds for AuxVerify. Let us consider a commitment to a group element $\hat{x}$ as $\hat{c} = \boldsymbol{e}\hat{x} + \hat{\boldsymbol{v}}r + \hat{\boldsymbol{w}}s$ with $r, s \leftarrow\$ \mathbb{Z}_p$, with corresponding auxiliary value $\mathsf{aux} = (\mathsf{aux}^1 = r\check{h}, \mathsf{aux}^2 = s\check{h})$. The check performed by AuxVerify is the following, where $\hat{x} \in \{\hat{g}, \hat{0}\}$ depends on the commitment and the auxiliary value to be checked.

$$\hat{c} \cdot (\check{h}, \check{h}) + \hat{\boldsymbol{v}} \cdot (-\mathsf{aux}^1, -\mathsf{aux}^1) + \hat{\boldsymbol{w}} \cdot (-\mathsf{aux}^2, -\mathsf{aux}^2) = (\hat{0}, \hat{x})^\top \cdot (\check{h}, \check{h})$$

Let us consider the left side of the above equation, we have:

$$[(\hat{0}, \hat{x}) + (r\xi\hat{g}, r\hat{g})^\top + (s\rho\xi\hat{g}, s\rho\hat{g})^\top] \cdot (\check{h}, \check{h}) +$$
$$(\xi\hat{g}, \hat{g})^\top \cdot (-r\check{h}, -r\check{h}) + (\rho\xi\hat{g}, \rho\hat{g})^\top \cdot (-s\check{h}, -s\check{h}) =$$
$$(\hat{0}, \hat{x})^\top \cdot (\check{h}, \check{h})$$

By observing that RandPr and AuxUpdate shift the commitments and the auxiliary values of the same randomness, it is straightforward to see that the first requirement of transformation completeness is also satisfied. Similarly, regarding the second requirement, PrAdd just replaces a commitment and corresponding auxiliary value with new ones that still pass the check performed by $\mathsf{AuxVerify}$. The same holds for the third requirement, where $\mathsf{PrExtend}$ just adds a new commitment with corresponding auxiliary value. $\square$

Lemma 10. *The proof system described above enjoys re-randomizable addition. (Def. 30).*

Proof. Let us point out the differences between the experiment executions with $b = 0$ and $b = 1$. In both cases, RandPr and $\mathsf{AuxUpdate}$ are executed to update both the proof and the auxiliary values before handling them to $\mathcal{A}$. Let us now consider what happens to the elements of Π and AUX before the above steps. When $b = 0$ both Π and AUX are freshly computed using the witness provided by $\mathcal{A}$. When $b = 1$ all proofs and commitments are just taken from the proof Π^* (provided by $\mathcal{A}$), excluding the ones related to variables b_{α_k}, $\hat{M}_{\alpha_k}$, x_{α_k} $y_{\alpha_k,j}$,for $j \in [J_{\alpha_k}]$, which are freshly generated.

Let us now focus on commitments $\hat{c}_{\hat{M}_i}$ to variables $\hat{M}_i$, with $i \in [n]$, contained in proof Π^*. It is straightforward to observe that after running $\mathsf{GS.RandPr}$, we get new commitments that are equally distributed to randomly chosen commitments to $\hat{M}_i$. The list AUX^* contains group elements satisfying the verification equations checked by $\mathsf{AuxVerify}$[33]. Auxiliary values are updated as random group elements of $\check{\mathbb{H}}$, using the same randomness previously used to re-randomize the commitments (which are elements of $\hat{\mathbb{G}}$). Due to transformation completeness, re-randomized commitments together with updated auxiliary values still satisfy such equation.

Therefore, the joint distribution of commitments and auxiliary values after re-randomization and update is equally distributed to randomly chosen commitments to $\hat{M}_i$ along with corresponding auxiliary values satisfying the verification equations checked by $\mathsf{AuxVerify}$.

Finally, as already shown in previous works [21,38,72], re-randomized proof elements (i.e, π values) are distributed as randomly chosen proof elements from the space of all valid proof elements, given that the commitments to the involved variables are fixed. Since the commitments

[33]We are guaranteed of that since otherwise $\mathcal{A}$ would not be admissible.

are fully re-randomized, the result is a randomly chosen proof given a fixed solution, along with openings of the randomly chosen commitments contained in the proof. This means that, for all equations involving any of the variables $\hat{M}_i$ with $i \in [n]$, the joint distribution of commitments, proofs elements, and AUX is identically distributed both when $b = 0$ and $b = 1$[34]. Finally, for all the equations not involving any of the variables $\hat{M}_i$ with $i \in [n]$, it suffices to reduce to the re-randomizability property of GS. $\square$

Lemma 11. *The proof system described above enjoys re-randomizable extension (Def. 31).*

Proof. The proof basically mirrors the one of re-randomizable addition. $\square$

Lemma 12. *The proof system described above enjoys extended witness indistinguishability (Def. 32).*

Proof. Notice that the proof Π contains a tuple of commitments and proof elements for each statement x_i with $i \in [n]$ (i.e., $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}i}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}i}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}i}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}i})$). Let us now consider the EWI experiment, where $\mathcal{A}$ chooses a statement $x = (k, x_1, \ldots, x_n)$ and two witnesses w_0 and w_1 such that $(x, w_0) \in \mathcal{R}_t$ and $(x, w_1) \in \mathcal{R}_t$. Let (Π_b, AUX_b) be the output of the challenger conditioned on the value of b. We argue that (Π_0, AUX_0) is indistinguishable from (Π_1, AUX_1).

For each of the tuples of the proof Π_b, we have that either $i \notin S$ or $i \in S$.

If $i \notin S$, AUX does not contain any information regarding the i-th tuple[35]. In this case, the i-th tuples when $b = 0$ and $b = 1$ are indistinguishable thanks to the WI of the underlying GS proof system.

If $i \in S$, we can distinguish two additional cases: (i) i is the index of a witness used both in w_0 and in w_1, and (ii) i is an index of a statement not used in both w_0 and w_1.

Let $i \in S$ be an index used both in w_0 and w_1. In this case, the value in $\hat{M}_i$ is $\hat{g}$. $\mathcal{A}$ obtained the auxiliary value associated with $\hat{M}_i$.

[34]In particular, note that after the invocation of PrAdd, all the proofs are accepting, including the one of equation $\mathcal{K}$.

[35]Notice that the randomness used in different commitments is always uniformly sampled.

We now consider all the elements of the i-th tuple and we argue that they are indistinguishable in both hybrids:

1. $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}i})$ are indistinguishable since $b_i = 1$ in both hybrids.

2. $(\boldsymbol{C}_{\boldsymbol{\mathcal{M}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}i})$ are indistinguishable since $\hat{M}_i = \hat{g}$, $b_i = 1$ in both hybrids;

3. Variables $\check{y}_{i,q,j}, \check{x}_{i,q,j}$ may differ in the two hybrids. However, $\mathcal{A}$ cannot distinguish $(\boldsymbol{C}_{\boldsymbol{\mathcal{Y}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}i})$ in the two hybrids. Indeed, the distribution of the proof elements is uniform over all possible terms satisfying the equation conditioned on the commitments, and all the commitments and auxiliary values are equally distributed in both hybrids.

4. Finally, $(\boldsymbol{C}_{\boldsymbol{\mathcal{X}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}i})$ do not involve any $\hat{M}_i$, and thus they are not related to AUX. They are indistinguishable thanks to the WI of the underlying GS proof system.

Whenever $i \in S$ is not used in both w_0 and w_1, we have that $\hat{M}_i = \hat{0}$ and AUX will contain openings to random commitments to $\hat{0}$. We notice that in the two hybrids, the entire i-th tuple is computed running the GS prover on the same inputs. Therefore, they are identically distributed.

There are two last element to take into account, namely $C_{\mathcal{K}}$ and $\pi_{\mathcal{K}}$. These are indistinguishable in both hybrids for the reasons previously argued in step 3. $\square$

Lemma 13. *The proof system described above enjoys adaptive extractable soundness (Def. 17).*

Proof. The algorithm Ext_1 just runs $\mathsf{GS.Ext}_1$. The algorithm Ext_2 runs $\mathsf{GS.Ext}_2$ for the equation $\mathcal{K}$. Due to the soundness of GS proofs, and thanks to the proofs of equations of type $\mathcal{M}$ and $\mathcal{B}$, we are guaranteed to extract, except with negligible probability, k values $\hat{M}_i = \hat{g}$. Let $A = \{\alpha_1, \ldots, \alpha_k : \hat{M}_{\alpha_1} = \ldots = \hat{M}_{\alpha_k} = \hat{g}\}$. Ext_2 runs $\mathsf{GS.Ext}_2$ for the k equations $\mathcal{X}_{\alpha_i}$ with $\alpha_i \in A$. Let w_i be the witness extracted from the proofs for equation $\mathcal{X}_{\alpha_i}$, with $\alpha_i \in A$. Due to the soundness of GS proofs, and thanks to the proofs of equations of type $\mathcal{Y}$ it holds that equations $\mathcal{X}_{\alpha_i}$ are the same as statement x_{α_i}, with $\alpha_i \in A$. Therefore, $w = \{(w_1, \alpha_1), \ldots, (w_k, \alpha_k)\}$ is such that $((k, x_1, \ldots, x_n), w) \in \mathcal{R}_t$. $\square$

Conclusion

In this thesis, we analyzed the provable security paradigm from three angles that we denoted as the Good, the Bad, and the Ugly. With *good* we denoted cryptographic protocols and primitives whose security is carefully formalized and argued according to the provable security paradigm. With *bad* we denoted the bad consequences of neglecting the design principles of provable security when building privacy-preserving complex systems. As a very recent example of “the bad”, we analysed the design flaws of the most widely deployed automatic contact tracing system (ACT) (i.e., GAEN/DP3T) which allow concerning privacy and integrity attacks. To this regard, our contribution was two-fold as we have both elucidated the feasibility of known and novel attacks - including the terrorist attack whose impact has been underestimated - and we have proposed two novel ACTs (i.e., Pronto-B2 and Pronto-C2) which are resilient to most of such attacks. With *ugly* we denoted situations in which the provable security paradigm could be deceiving, meaning that the approach is followed when proposing a new cryptographic scheme/protocol, but the construction may be insecure when deployed in practice. To this regard, we considered the problem of both faulty security proofs and inadequate definitions that were present in recent publications in the area of threshold cryptography. In particular, we focused on proofs over threshold relations, threshold ring signatures, and extendable threshold ring signatures. We overcame the issues of previous works by proposing stronger definitions, new constructions, and revisited security proofs. Additionally, our new constructions improve the previous ones in terms of efficiency, security, and/or features.

In the area of proofs over threshold relations, a recent line of work [1,124] focused on proving statements that consist of several theorems composed in conjunctive normal form (CNF), that is a conjunction of several disjunctive statements. A proof for a CNF statement can be very straightforwardly obtained by running several $(1, n)$-PTRs in parallel. Nevertheless, this simple approach neglects any saving (of computation or communication) that could be obtained exploiting the fact that the same theorems may be repeated in multiple clauses. Recent works exploited this fact by modelling the statement in CNF as a directed acyclic graph (DAG), and then running a proof about the DAG which is much smaller than - but logically equivalent to - the initial statement. An interesting open question is to assess whether stacking techniques can be used to obtain further improvements, and to analyze the case of conjunctions of (k, n)-PTRs, which has not been the focus of any prior work.

There is a consistent body of literature works that has analyzed several different aspects of ring signatures. Some recent works have focused on obtaining compact ring signatures in the plain model [17], and even from post-quantum secure assumptions [40]. Another recent work also answered the theoretical question of building a ring signature scheme by composing regular signature schemes in a black-box way [29]. Being extendable threshold ring signatures a very recent notion, such aspects have not been explored in this context yet and are an interesting direction for future work.

Bibliography

[1] Masayuki Abe, Miguel Ambrona, Andrej Bogdanov, Miyako Ohkubo, and Alon Rosen. Acyclicity programming for sigma-protocols. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 435–465, Raleigh, NC, USA, November 8–11, 2021. Springer, Heidelberg, Germany.

[2] Carlos Aguilar Melchor, Pierre-Louis Cayrel, and Philippe Gaborit. A new efficient threshold ring signature scheme based on coding theory. In Johannes Buchmann and Jintai Ding, editors, *Post-quantum cryptography, second international workshop, PQCRYPTO 2008*, pages 1–16, Cincinnati, Ohio, United States, October 17–19, 2008. Springer, Heidelberg, Germany.

[3] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco, and Jason Yellick. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, pages 30:1–30:15. ACM, 2018.

[4] Apple and Google. Apple and Google's exposure notification system. 2020. https://www.apple.com/covid19/contacttracing.

[5] Diego F. Aranha, Mathias Hall-Andersen, Anca Nitulescu, Elena Pagnin, and Sophia Yakoubov. Count Me In! Extendability for Threshold Ring Signatures. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 379–406. Springer, Heidelberg, Germany, 2022.

[6] Wikipedia Article. Pan-european privacy-preserving proximity tracing. *Wikipedia*, Oct 2022. https://en.wikipedia.org/wiki/Pan-European_Privacy-Preserving_Proximity_Tracing.

[7] Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing proofs of k-out-of-n partial knowledge. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 65–91, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

[8] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. Compressed Σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 526–556, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany.

[9] Benedikt Auerbach, Suvradip Chakraborty, Karen Klein, Guillermo Pascual-Perez, Krzysztof Pietrzak, Michael Walter, and Michelle Yeo. Inverse-sybil attacks in automated contact tracing. In Kenneth G. Paterson, editor, *Topics in Cryptology – CT-RSA 2021*, volume 12704 of *Lecture Notes in Computer Science*, pages 399–421, Virtual Event, May 17–20, 2021. Springer, Heidelberg, Germany.

[10] Gennaro Avitabile, Vincenzo Botta, and Dario Fiore. Extendable threshold ring signatures with enhanced anonymity. Cryptology ePrint Archive, Paper 2022/1568, 2022. https://eprint.iacr.org/2022/1568.

[11] Gennaro Avitabile, Vincenzo Botta, Daniele Friolo, and Ivan Visconti. Efficient proofs of knowledge for threshold relations. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022: 27th European Symposium on Research in Computer Security, Part III*, volume 13556 of *Lecture Notes in Computer Science*, pages 42–62, Copenhagen, Denmark, September 26–30, 2022. Springer, Heidelberg, Germany.

[12] Gennaro Avitabile, Vincenzo Botta, Vincenzo Iovino, and Ivan Visconti. Towards defeating mass surveillance and sars-cov-2: The pronto-c2 fully decentralized automatic contact tracing system. Cryptology ePrint Archive, Paper 2020/493, 2020. https://eprint.iacr.org/2020/493.

[13] Gennaro Avitabile, Vincenzo Botta, Vincenzo Iovino, and Ivan Visconti. Towards Defeating Mass Surveillance and SARS-CoV-2: The Pronto-C2 Fully Decentralized Automatic Contact Tracing System. In *NDSS CoronaDef Workshop on Secure IT Technologies against COVID-19*, pages 45–52, 2021.

[14] Gennaro Avitabile, Vincenzo Botta, Vincenzo Iovino, and Ivan Visconti. Privacy and integrity threats in contact tracing systems and their mitigations. *IEEE Internet Computing*, 2022. To appear.

[15] Gennaro Avitabile, Daniele Friolo, and Ivan Visconti. Terrorist attacks for fake exposure notifications in contact tracing systems. Cryptology ePrint Archive, Paper 2020/1150, 2020. https://eprint.iacr.org/2020/1150.

[16] Gennaro Avitabile, Daniele Friolo, and Ivan Visconti. Terrorist attacks for fake exposure notifications in contact tracing systems. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS*

21: 19th International Conference on Applied Cryptography and Network Security, Part I, volume 12726 of *Lecture Notes in Computer Science*, pages 220–247, Kamakura, Japan, June 21–24, 2021. Springer, Heidelberg, Germany.

[17] Michael Backes, Nico Döttling, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Ring signatures: Logarithmic-size, no setup - from standard assumptions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 281–311, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[18] Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 92–122, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

[19] Lars Baumgärtner, Alexandra Dmitrienko, Bernd Freisleben, Jonas Höchst, Mira Mezini, Markus Miettinen, Thien Duc Nguyen, Alvar Penning, Filipp Roos, Ahmad-Reza Sadeghi, Michael Schwarz, and Christian Uhl. Mind the GAP: Security & privacy risks of contact tracing apps. In *TrustCom 2020, Security Track*, pages 458–467, 2020.

[20] Johannes K Becker, David Li, and David Starobinski. Tracking anonymized bluetooth devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65, 2019.

[21] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.

[22] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 701–732, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[23] Slim Bettaieb and Julien Schrek. Improved lattice-based threshold ring signature scheme. In Philippe Gaborit, editor, *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013*, pages 34–51, Limoges, France, June 4–7, 2013. Springer, Heidelberg, Germany.

[24] Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 464–492, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.

[25] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.

[26] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[27] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

[28] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable

ring signatures based on DDH. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015: 20th European Symposium on Research in Computer Security, Part I*, volume 9326 of *Lecture Notes in Computer Science*, pages 243–265, Vienna, Austria, September 21–25, 2015. Springer, Heidelberg, Germany.

[29] Pedro Branco, Nico Döttling, and Stella Wohnig. Universal ring signatures in the standard model. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 249–278. Springer, 2022.

[30] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

[31] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 351–368, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany.

[32] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, jul 2004.

[33] Ran Canetti and Shafi Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 90–106, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

[34] Ran Canetti, Yael Tauman Kalai, Anna Lysyanskaya, Ronald L. Rivest, Adi Shamir, Emily Shen, Ari Trachtenberg, Mayank Varia, and Daniel J. Weitzner. Privacy-preserving automated exposure notification. Cryptology ePrint Archive, Paper 2020/863, 2020. https://eprint.iacr.org/2020/863.

[35] Alessandro De Carli, Muriel Figueredo Franco, A. Gassmann, Christian Killer, Bruno Rodrigues, Eder J. Scheid, D. Schoenbaechler, and Burkhard Stiller. Wetrace - A privacy-preserving mobile COVID-19 tracing approach and application. *CoRR*, abs/2004.08812, 2020.

[36] Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. https://eprint.iacr.org/2022/975.

[37] Justin Chan, Dean P. Foster, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham M. Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Sudheesh Singanamalla, Jacob E. Sunshine, and Stefano Tessaro. PACT: privacy sensitive protocols and mechanisms for mobile contact tracing. *CoRR*, abs/2004.03544, 2020. https://arxiv.org/abs/2004.03544.

[38] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 281–300, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[39] 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, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

[40] Rohit Chatterjee, Sanjam Garg, Mohammad Hajiabadi, Dakshita Khurana, Xiao Liang, Giulio Malavolta, Omkant Pandey, and Sina Shiehian. Compact ring signatures from learning with

errors. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 282–312, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

[41] David Chaum. Blind signature system. In David Chaum, editor, *Advances in Cryptology – CRYPTO'83*, page 153, Santa Barbara, CA, USA, 1983. Plenum Press, New York, USA.

[42] David Chaum. Blind signature systems. U.S. Patent #4,759,063, July 1988.

[43] Jiangshan Chen, Yupu Hu, Wen Gao, and Hong-Liang Li. Lattice-based Threshold Ring Signature with Message Block Sharing. *KSII*, 13(2):1003–1019, 2019.

[44] Sherman S. M. Chow, Victor K.-W. Wei, Joseph K. Liu, and Tsz Hon Yuen. Ring signatures without random oracles. In Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Lin, Shiuhpyng Shieh, and Sushil Jajodia, editors, *ASIACCS 06: 1st ACM Symposium on Information, Computer and Communications Security*, pages 297–302, Taipei, Taiwan, March 21–24, 2006. ACM Press.

[45] 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, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 63–92, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[46] Andrew Clement, Jilian Harkness, George Rain, and Laura Tribe. Snowden surveillance archive. 2020. https://snowdenarchive.cjfe.org/greenstone/cgi-bin/library.cgi.

[47] 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*,

volume 1462 of *Lecture Notes in Computer Science*, pages 424–441, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

[48] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Heidelberg, Germany.

[49] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

[50] Noel Danz, Oliver Derwisch, Anja Lehmann, Wenzel Puenter, Marvin Stolle, and Joshua Ziemann. Provable security analysis of decentralized cryptographic contact tracing. Cryptology ePrint Archive, Report 2020/1309, 2020. https://ia.cr/2020/1309.

[51] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[52] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In Matt Blaze, editor, *USENIX Security 2004: 13th USENIX Security Symposium*, pages 303–320, San Diego, CA, USA, August 9–13, 2004. USENIX Association.

[53] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer*

Science, pages 609–626, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[54] Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 630–649, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[55] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 67–88, Bogota, Colombia, June 5–7, 2019. Springer, Heidelberg, Germany.

[56] European Commission. Guidance on apps supporting the fight against COVID 19 pandemic in relation to data protection. *Official Journal of the European Union*, 2020. https://ec.europa.eu/info/sites/default/files/5_en_act_part1_v3.pdf.

[57] Antonio Faonio, Dario Fiore, Luca Nizzardo, and Claudio Soriente. Subversion-resilient enhanced privacy ID. In Steven D. Galbraith, editor, *Topics in Cryptology – CT-RSA 2022*, volume 13161 of *Lecture Notes in Computer Science*, pages 562–588, Virtual Event, March 1–2, 2022. Springer, Heidelberg, Germany.

[58] 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*, pages 308–317, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press.

[59] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall,

and Christophe Fraser. Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. *Science*, 368(6491):eabb6936, 2020.

[60] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

[61] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. Applications of Algebra to Cryptography.

[62] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. Stacking Sigmas: A Framework to Compose Σ-Protocols for Disjunctions. Cryptology ePrint Archive, Report 2021/422, 2021. Version of 09 November 2021 https://eprint.iacr.org/archive/2021/422/20211109:180510.

[63] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. Stacking Sigmas: A Framework to Compose Σ-Protocols for Disjunctions. Cryptology ePrint Archive, Report 2021/422, 2022. Version of 20 October 2022 https://eprint.iacr.org/archive/2021/422/20221020:032739.

[64] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. Stacking sigmas: A framework to compose Σ-protocols for disjunctions. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 458–487, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.

[65] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994.

[66] Google. Google: Exposure notification reference key server. 2020. https://google.github.io/exposure-notifications-server/.

[67] Google and Apple. Exposure notification cryptography specification. 2020. https://blog.google/documents/69/Exposure_Notification_-_Cryptography_Specification_v1.2.1.pdf.

[68] Google and Apple. Google and Apple's exposure notification system. 2020. https://www.apple.com/covid19/contacttracing.

[69] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459, Shanghai, China, December 3–7, 2006. Springer, Heidelberg, Germany.

[70] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 253–280, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[71] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[72] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[73] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *Advances in Cryptology – EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128, Davos, Switzerland, May 25–27, 1988. Springer, Heidelberg, Germany.

[74] Abida Haque, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. Logarithmic-size (linkable) threshold ring signatures in the plain model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 437–467. Springer, Heidelberg, Germany, 2022.

[75] Abida Haque and Alessandra Scafuro. Threshold ring signatures: New definitions and post-quantum security. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 423–452, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.

[76] David Heath and Vladimir Kolesnikov. Stacked garbling for disjunctive zero-knowledge proofs. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 569–598, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

[77] David Heath, Vladimir Kolesnikov, and Stanislav Peceny. Garbling, stacked and staggered - faster k-out-of-n garbled function evaluation. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 245–274, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany.

[78] Immuni Team. Immuni's high-level description. https://github.com/immuni-app/immuni-documentation, 2020. Accessed: 2020-08-23.

[79] Inria PRIVATICS Team. DESIRE: A Third Way for a European Exposure Notification System. https://github.com/3rd-ways-for-EU-exposure-notification/project-DESIRE/blob/master/DESIRE-specification-EN-v1_0.pdf, 2020. Accessed: 2020-06-03.

[80] Vincenzo Iovino, Serge Vaudenay, and Martin Vuagnoux. On the effectiveness of time travel to inject COVID-19 alerts. In Kenneth G. Paterson, editor, *Topics in Cryptology – CT-RSA 2021*, volume 12704 of *Lecture Notes in Computer Science*, pages 422–443, Virtual Event, May 17–20, 2021. Springer, Heidelberg, Germany.

[81] Neal Koblitz and Alfred Menezes. Another look at "provable security". Cryptology ePrint Archive, Paper 2004/152, 2004. https://eprint.iacr.org/2004/152.

[82] Franck Legendre, Mathias Humbert, Alain Mermoud, and Vincent Lenders. Contact tracing: An overview of technologies and cyber risks. *CoRR*, abs/2007.02806, 2020. https://arxiv.org/abs/2007.02806.

[83] Dough Leith and Stephen Farrell. Testing apps for COVID-19 tracing (TACT). https://down.dsg.cs.tcd.ie/tact/, 2020. Accessed: 2020-08-23.

[84] Douglas J. Leith and Stephen Farrell. Coronavirus contact tracing: evaluating the potential of using bluetooth received signal strength for proximity detection. *Comput. Commun. Rev.*, 50(4):66–74, 2020.

[85] Zhen Liu, Khoa Nguyen, Guomin Yang, Huaxiong Wang, and Duncan S. Wong. A lattice-based linkable ring signature supporting stealth addresses. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019: 24th European Symposium on Research in Computer Security, Part I*, volume 11735 of *Lecture Notes in Computer Science*, pages 726–746, Luxembourg, September 23–27, 2019. Springer, Heidelberg, Germany.

[86] Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 110–130, Bogota, Colombia, June 5–7, 2019. Springer, Heidelberg, Germany.

[87] Luciano Maino and Chloe Martindale. An attack on sidh with arbitrary starting curve. Cryptology ePrint Archive, Paper 2022/1026, 2022. https://eprint.iacr.org/2022/1026.

[88] Deepak Maram, Harjasleen Malvai, Fan Zhang, Nerla Jean-Louis, Alexander Frolov, Tyler Kell, Tyrone Lobban, Christine Moy, Ari Juels, and Andrew Miller. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1348–1366, 2021.

[89] Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *AFRICACRYPT 09: 2nd International Conference on Cryptology in Africa*, volume 5580 of *Lecture Notes in Computer Science*, pages 272–286, Gammarth, Tunisia, June 21–25, 2009. Springer, Heidelberg, Germany.

[90] Alexander Munch-Hansen, Claudio Orlandi, and Sophia Yakoubov. Stronger notions and a more efficient construction of threshold ring signatures. In Patrick Longa and Carla Ràfols, editors, *Progress in Cryptology - LATINCRYPT 2021: 7th International Conference on Cryptology and Information Security in Latin America*, volume 12912 of *Lecture Notes in Computer Science*, pages 363–381, Bogotá, Colombia, October 6–8, 2021. Springer, Heidelberg, Germany.

[91] Moni Naor. Deniable ring authentication. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 481–498, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

[92] Anca Nitulescu. zk-SNARKs: A Gentle Introduction. https://www.di.ens.fr/~nitulesc/files/Survey-SNARKs.pdf, 2020.

[93] Takeshi Okamoto, Raylin Tso, Michitomo Yamaguchi, and Eiji Okamoto. A k-out-of-n ring signature with flexible participation for signers. Cryptology ePrint Archive, Paper 2018/728, 2018. https://eprint.iacr.org/2018/728.

[94] PACT Team. Decentralized privacy-preserving proximity tracing. 2020. https://pact.mit.edu/wp-content/uploads/2020/04/The-PACT-protocol-specification-ver-0.1.pdf.

[95] Albrecht Petzoldt, Stanislav Bulygin, and Johannes Buchmann. A multivariate based threshold ring signature scheme. *Appl. Algebra Eng. Commun. Comput.*, 24(3-4):255–275, 2013.

[96] Krzysztof Pietrzak. Delayed authentication: Preventing replay and relay attacks in private contact tracing. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *Progress in Cryptology - INDOCRYPT 2020: 21st International Conference in Cryptology in India*, volume 12578 of *Lecture Notes in Computer Science*, pages 3–15, Bangalore, India, December 13–16, 2020. Springer, Heidelberg, Germany.

[97] Benny Pinkas and Eyal Ronen. Hashomer – privacy-preserving bluetooth based contact tracing scheme for hamagen. In *Workshop on Secure IT Technologies against COVID-19 (CoronaDef), 21 February*, 2021. ISBN: 1-891562-72-X.

[98] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany.

[99] Adam Krellenstein Rosario Gennaro and James Krellenstein. Exposure notification system may allow for large-scale voter suppression. https://static1.squarespace.com/static/5e937

afbfd7a75746167b39c/t/5f47a87e58d3de0db3da91b2/1598531714869/Exposure_Notification.pdf, 2020. Accessed: 2020-08-23.

[100] Antonio Russo, Antonio Fernández Anta, María Isabel González Vasco, and Simon Pietro Romano. Chirotonia: A scalable and secure e-voting framework based on blockchains and linkable ring signatures. In *2021 IEEE International Conference on Blockchain (Blockchain)*, pages 417–424, 2021.

[101] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[102] Otto Seiskari. Contact Tracing BLE sniffer PoC. https://github.com/oseiskar/corona-sniffer, 2020. Accessed: 2020-06-02.

[103] Semaphore Team. Semaphore. https://semaphore.appliedzkp.org/, 2020. Accessed: 2020-09-15.

[104] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *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 207–220. Springer, 2000.

[105] Swiss Federal Office of Public Health. New coronavirus: Swisscovid app and contact tracing. https://www.bag.admin.ch/bag/en/home/krankheiten/ausbrueche-epidemien-pandemien/aktuelle-ausbrueche-epidemien/novel-cov/swisscovid-app-und-contact-tracing/datenschutzerklaerung-nutzungsbedingungen.html#-11360452, 2020. Accessed: 2020-08-23.

[106] TU DARMSTADT SYSTEM SECURITY LAB, CYSEC. TraceCORONA: Anonymous distributed contact tracing for

pandemic response. https://tracecorona.net/, 2020. Accessed: 2020-09-05.

[107] Qiang Tang. Privacy-preserving contact tracing: current solutions and open questions. *CoRR*, abs/2004.06818, 2020. https://doi.org/10.48550/arXiv.2004.06818.

[108] TCNCoalition. TCN Protocol. https://github.com/TCNCoalition/TCN#the-tcn-protocol, 2020. Accessed: 2020-05-03.

[109] DP3T Team. DESIRE: A Practical Assessment. https://github.com/DP-3T/documents/blob/master/Security%20analysis/DESIRE%20-%20A%20Practical%20Assessment.pdf, 2020. Accessed: 2020-06-01.

[110] DP3T Team. Privacy and Security Risk Evaluation of Digital Proximity Tracing Systems. https://github.com/DP-3T/documents/blob/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tracing%20Systems.pdf, 2020. Accessed: 2020-04-21.

[111] DP3T Team. Response to 'Analysis of DP3T: Between Scylla and Charybdis'. https://github.com/DP-3T/documents/blob/master/Security%20analysis/Response%20to%20'Analysis%20of%20DP3T'.pdf, 2020. Accessed: 2020-04-23.

[112] DP3T Team. Secure upload authorisation for digital proximity tracing. https://github.com/DP-3T/documents/blob/master/DP3T%20-%20Upload%20Authorisation%20Analysis%20and%20Guidelines.pdf, 2020. Accessed: 2020-05-03.

[113] Sri Aravinda Krishnan Thyagarajan, Giulio Malavolta, Fritz Schmid, and Dominique Schröder. Verifiable timed linkable ring signatures for scalable payments for monero. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022: 27th European Symposium on Research in Computer Security, Part II*, volume 13555 of *Lecture Notes in Computer Science*, pages 467–486, Copenhagen, Denmark, September 26–30, 2022. Springer, Heidelberg, Germany.

[114] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James R. Larus, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth G. Paterson, Srdjan Capkun, David A. Basin, Jan Beutel, Dennis Jackson, Marc Roeschlin, Patrick Leu, Bart Preneel, Nigel P. Smart, Aysajan Abidin, Seda Gurses, Michael Veale, Cas Cremers, Michael Backes, Nils Ole Tippenhauer, Reuben Binns, Ciro Cattuto, Alain Barrat, Dario Fiore, Manuel Barbosa, Rui Oliveira, and José Pereira. Decentralized privacy-preserving proximity tracing. 2020. https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf.

[115] Serge Vaudenay. Analysis of DP3T. Cryptology ePrint Archive, Paper 2020/399, 2020. https://eprint.iacr.org/2020/399.

[116] Serge Vaudenay. Centralized or decentralized? the contact tracing dilemma. Cryptology ePrint Archive, Paper 2020/531, 2020. https://eprint.iacr.org/2020/531.

[117] Serge Vaudenay and Martin Vuagnoux. Analysis of swisscovid. https://lasec.epfl.ch/people/vaudenay/swisscovid/swisscovid-ana.pdf, 2020. Accessed: 2020-08-23.

[118] Serge Vaudenay and Martin Vuagnoux. Swisscovid in the perspective of its goals. *Digital Threats*, 3(3), feb 2022.

[119] Ivan Visconti. Zero-knowledge proofs. In D. Pointcheval, editor, *Asymmetric Cryptography: Primitives and Protocols*, chapter 3, pages 63–80. Wiley, 2022.

[120] Zachary J. Williamson. Aztec. https://github.com/AztecProtocol/AZTEC/blob/master/AZTEC.pdf, 2018. Accessed: 2020-09-15.

[121] H. Yang. *EC Cryptography Tutorials - Herong's Tutorial Examples*. Herong's Tutorial Examples. Herong Yang, 2019.

[122] Tsz Hon Yuen, Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Efficient Linkable and/or Threshold Ring

Signature Without Random Oracles. *The Computer Journal*, 56(4):407–421, 2013.

[123] Tsz Hon Yuen, Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Efficient linkable and/or threshold ring signature without random oracles. *Comput. J.*, 56(4):407–421, 2013.

[124] Gongxian Zeng, Junzuo Lai, Zhengan Huang, Yu Wang, and Zhiming Zheng. Dag-Σ: A dag-based sigma protocol for relations in CNF. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part II*, volume 13792 of *Lecture Notes in Computer Science*, pages 340–370. Springer, 2022.

[125] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 270–282, Vienna, Austria, October 24–28, 2016. ACM Press.

[126] Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. DECO: Liberating web data using decentralized oracles for TLS. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1919–1938, Virtual Event, USA, November 9–13, 2020. ACM Press.

[127] Fangguo Zhang and Kwangjo Kim. ID-based blind signature and ring signature from pairings. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany.

[128] Guomin Zhou, Peng Zeng, Xiaohui Yuan, Siyuan Chen, and Kim-Kwang Raymond Choo. An Efficient Code-Based Threshold Ring Signature Scheme with a Leader-Participant Model. *Secur. Commun. Networks*, 2017:1–7, 2017.

[129] Guomin Zhou, Peng Zeng, Xiaohui Yuan, Siyuan Chen, and Kim-Kwang Raymond Choo. An efficient code-based threshold ring signature scheme with a leader-participant model. *Secur. Commun. Networks*, 2017:1915239:1–1915239:7, 2017.

[130] ZkDAI Team. Zkdai. https://github.com/atvanguard/ethsingapore-zk-dai, 2020. Accessed: 2020-09-15.

[131] ZoKrates Team. Zokrates. https://zokrates.github.io/, 2020. Accessed: 2020-09-15.