UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA “RENATO M. CAPOCELLI”

DOTTORATO DI RICERCA IN INFORMATICA

XIII CICLO

TESI DI DOTTORATO IN INFORMATICA

ID-Based Key Agreement for WANETs

Candidato

Francesco Rossi

Coordinatore

Prof. Giuseppe Persiano

Tutor

Prof. Alfredo De Santis

Co-Tutor

Prof. Giovanni Schmid

Anno Accademico 2013/2014

Declaration

I herewith declare that i have produced this work without the prohibited assistance of third parties and without making use of aids other than those specified. Notions taken over directly or indirectly from other sources have been identified as such. Some results in this thesis are present in the following papers:

Rossi, F. and Schmid, G. (2015). Implementing identity-based key agreement in embedded devices. In *PECCS 2015 - Proceedings of the 5st International Conference on Pervasive and Embedded Computing and Communication Systems, Angers, France, 11-13 February, 2015*, pages 117–123

Schmid, G. and Rossi, F. (2012). QR code-based identification with mobile devices. In *PECCS 2012 - Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems, Rome, Italy, 24-26 February, 2012*, pages 79–86

Schmid, G. and Rossi, F. (2011). Secure ad-hoc routing through a-codes. In *PECCS 2011 - Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems, Vilamoura, Algarve, Portugal, 5-7 March, 2011*, pages 151–156

Dedication and Acknowledgments

I dedicate this thesis to Imma and to my family, who made it possible and supported me throughout these years.

I am heartily thankful to Prof. Giovanni Schmid, whose guidance accompanied me from the early stages of my writing until the very end. This thesis would not have been possible without his commitment, suggestions and feedbacks.

I am indebted to Francesco Maione, whose help has been essential to my writing.

Finally, a last thank to Prof. Alfredo De Santis and to my friends of University of Salerno, it has been an honor for me to meet all of you.

Contents

1. Introduction 8

2. Authentication in WANETs 12

2.1 Security attacks 14

2.2 Authentication mechanisms 17

2.2.1 Certificate-Based key distribution 18

2.2.2 ID-based key distribution 19

2.3 Key establishment 20

2.4 Dieffie-Hellman protocol 21

3. Theoretical background 24

3.1 Finite fields and discrete logarithm systems 24

3.2 Elliptic curves 26

3.3 Bilinear pairings and co-Diffie-Hellman problems 27

3.3.1 Bilinear pairings from the tate pairings 29

3.4 ID-based cryptography and hashing function algorithms 31

3.5 Point compression 34

4. ID-based signature 35

4.1 IBS-1 and IBS-2 schemes 37

4.2 GS-IBS scheme 42

4.3 Comparison of computational costs 43

5. ID-based key agreement 45

5.1 eFG protocol 46

5.2 Extending the DH-protocol to n parties 50

5.3 Constant round protocols 52

5.4 GKA protocols 54

5.4.1 Protocol GKA v1 56

5.4.2 Protocol GKA v2 59

5.4.3 Comparison of computational costs 61

5.4.4 Run example 62

5.4.5 Correctness and security 64

5.5 GKA dynamic procedures 69

5.5.1 Group creation and key updating 70

5.5.2 Group joining and leaving 74

6. Java for ID-based key agreement 80

6.1 JIKA software architecture 81

6.2 Implementation of elliptic curves and tate pairing 83

6.3 Performance evaluation 85

6.3.1 Enrollment 85

6.3.2 Signature schemes 87

6.3.3 Key agreement protocols 88

eFG protocol 88

GKA protocols 91

Conclusions 99

Bibliography 101

List of Figures

2.1 A WANET organized in flat and hierarchical topologies respectively. Red circles and orange nodes represent different clusters with their cluster heads. 13
2.2 Instance of unicast, multicast and broadcast communications. The red and green colors indicate sender and receiver nodes respectively. . 17
2.3 DH protocol. 22
2.4 Man in the middle attack against DH protocol. 22
2.5 STS protocol. 23

6.1 JIKA framework software architecture. 81
6.2 GKA v1 computing times versus involved parties on the PC platform using elliptic curves ssTate160 and nssTate160, respectively. 93
6.3 GKA v2 computing times versus involved parties on the PC platform using elliptic curves ssTate160 and nssTate160, respectively. 94
6.4 GKA v1 computing times versus involved parties on Raspberry PI using elliptic curves ssTate160 and nssTate160, respectively. 95
6.5 GKA v2 computing times versus involved parties on Raspberry PI using elliptic curves ssTate160 and nssTate160, respectively. 96

List of Algorithms

Algorithm **Page**

MapToGroup 34

ID-based signature schemes

Setup 36

KeyGen 37

IBS-1 38

IBS-2 40

GS-IBS 42

Key agreement protocols

eFG Setup 46

eFG KeyGen 47

eFG 48

GKA v1 57

GKA v2 59

Introduction

In the last few years, the research in the area of wireless ad hoc networks (WANETs) has steep growth. The increasing interest about ad hoc networks is due to some key features not owned by traditional networks such as nodes mobility, network self-organization and the ability to rely on infrastructure-less setup. Due to the absence of a fixed infrastructure, WANETs can be used in many application scenarios. A set of possible applications includes the following:

- *Health Care*: Wearable sensors can help patients by providing healthcare services such as medical monitoring and communication systems with healthcare provider in emergency situations. Moreover, they can provide methods to remotely access to physiological informations, thus allowing people control continually patients conditions, improving their life quality.

- *Environmental monitoring*: Natural disasters are increasing due to climate changes and pollution. Sensor nodes can be used to monitor habitats, air conditions, geographic areas, animal habits and many other phenomena. WANETs equipped with sensor nodes can be deployed in inaccessible areas, in order to observe phenomena locally and transmit data to sync nodes, that can perform different operations such as collecting data or activating alarms.

- *Rescue operations*: WANETs have the capability of managing data in real-time and sending them with a minimum delay. Hence, these systems can be used to support rescue operations and save people life.

- *Military*: WANETs can be used to establish communications among soldiers and vehicles for tactical operations in battlefields.

The open nature of the communication channel exposes WANETs to a great number of security threats. For example, malicious adversaries can eavesdrop messages, impersonate legitimate nodes, manipulates routing paths and cause many others misbehaviours. The security of WANETs hinges on node authentication, which by mean of Cryptography can be obtained through key distribution mechanisms. Unfortunately, traditional key distribution mechanisms can be inapplicable for WANETs. For example, certificated-based approaches require high computational capabilities for verification, update and revocation of public key certificates (PKCs), thus being impractical for resource constrained nodes. Alternatively, identity-based (ID-based) mechanisms use identity attributes instead of PKCs requiring less resources consuming than certificated-based approaches.
Generally, the design of cryptographic protocols must necessary take into account different efficiency parameters such as number of operations, use of network bandwidth and energy consumption. In particular, power conservation takes on additional importance when nodes are equipped with external batteries having limited lifetime.
WANET applications often require the establishment of session keys, that will be used for encryption, authentication and others cryptographic purposes. Key agreement protocols provide session keys thorough contributory approaches, which means that nodes can't predetermine keys and they give the same contribution to carrying out the protocol. This way, it's possible to increase node lifetime, since the resource consumption is equally distributed among all nodes.

Our Contribution

In this thesis we present a cryptographic framework for WANETs, named *JIKA* (Java framework for ID-based key agreement). The JIKA framework takes advantage of the portability of Java programming language, and due to a modular software architecture, it can be easily extended to encompass many cryptographic schemes and protocols.

JIKA simulates a key generation center (KGC) and offers an ID-based key distribution service for signature schemes and key agreement protocols. It requires each node storing in memory only long-term keying materials and system public parameters. Moreover, JIKA makes use of elliptic curve cryptography (ECC) which allows fast computations, small key size and short signatures of messages. ECC operates on groups of points where the elliptic curve discrete logarithm problem (ECDLP) is hard to solve. Whereas sub-exponential algorithms are known to solve the discrete logarithm problem (DLP) problem, only exponential algorithms are known to solve the ECDLP.

JIKA includes two new ID-based signature schemes *IBS-1* and *IBS-2*, that were derived from the BLS scheme of [Boneh et al., 2004] and the ZSS scheme of [Zhang et al., 2004]. Thanks to pairing-based cryptography, both schemes get shorter signatures if compared to other signature algorithms over elliptic curves, e.g. the signature standard algorithm ECDSA [Johnson et al., 2001]. JIKA includes also an ID-based two-party key agreement protocol, named *eFG*. This protocol, derived from [Fiore and Gennaro, 2010], requires just one round and it appears very suitable for ad hoc networks where nodes need to establish session keys quickly and at low computational cost. As another contribution, JIKA includes new group key agreement protocols, named *GKA*. They are full-contributory protocols that offer implicit key authentication through the ID-based signature schemes described above, at the cost of just two rounds. GKA provide resilience against passive and active attacks performed by an adversary which is not a legitimate node. Related to GKA protocols, we propose a set of procedures for dynamic group management. These procedures allow for group creation, key updating, node joining and node leaving, respectively. In order to measure the performance of the proposed algorithms, we implemented in JIKA some notable signature schemes and key agreement protocols [Johnson et al., 2001],[Gentry and Silverberg, 2002],[Rogaway et al., 2001] and [Burmester and Desmedt, 1994]. Moreover, in order to emulate executions on a WANET, we run the above algorithms on the embedded devices Raspberry PI [Upton and Halfacree, 2013]. Indeed, Raspberry PI devices support standard wireless adapters that can be used to setup a WANET through a suitable configuration [Debian, 2015]. The same tests were executed on a Personal Computer (PC) platform, so as to compare results between devices with very different computational resources.

Roadmap

This thesis is organized as follows. In the next chapter, Chapter 2, we introduce WANETs and discuss briefly their security issues. Moreover, we overview basic results concerning entity authentication and key establishment that are useful for the sequel. Chapter 3 is related to theoretical background about hardness assumptions, elliptic curve arithmetic and ID-based schemes. Chapter 4 is related to schemes *IBS-1* and *IBS-2*. We analyze their computational costs and prove their correctness and security. Chapter 5 concerns instead ID-based key agreement protocols. We discuss the two-party key agreement protocol *eFG* and the group key agreement protocols *GKA*, proving their correctness and security. Moreover, we introduce a set of procedures for the management of dynamic groups of parties. Finally, in Chapter 6 we illustrate the software architecture of JIKA and present the tests performed on a Raspberry PI and PC platform.

Authentication in WANETs

A WANET is a collection of wireless mobile nodes that dynamically form a network without the aid of any infrastructure [Sarkar et al., 2007]. WANETs can be arranged in *hierarchical* and *flat* topologies. The hierarchical topology allows to organize the network in two or more clusters and communications inside a cluster are routed by cluster heads. Unfortunately, if a cluster head goes down, a complete section of the network will be unreachable. Although hierarchical topologies provide a good scalability, the cluster heads create single points of failure. The flat topology consists of nodes which have the same purposes and offer the same functionalities. These organization allows for a batter distribution of workload and it aims in to reduce resources consumption.

The open nature of wireless communication channel makes WANETs vulnerable to different kind of security attacks. Generally, we classified *passive* and *active* attacks. A passive attack is mainly against data confidentiality, it's performed by an adversary who has eavesdropped network messages to gain sensitive informations. Instead, an active attack involve an adversary who attempts to modify or inject network messages. For example, it could initiate new protocol instances or impersonates others nodes. The adversaries can also classified in *outsider* and *insider*. An outsider adversary is an unauthorized node without special privileges or knowledges of the network. Instead, an insider adversary is a legitimate node of network with additional reserved informations such as secret keys and routing paths [Pathan, 2010].

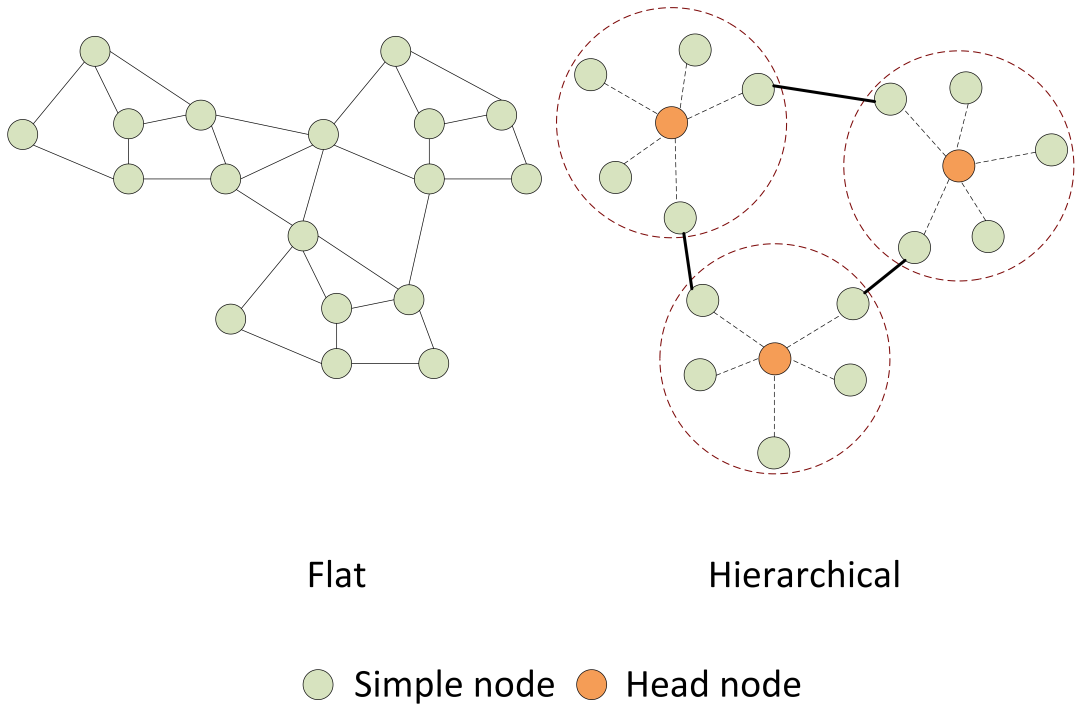


Fig. 2.1: A WANET organized in flat and hierarchical topologies respectively. Red circles and orange nodes represent different clusters with their cluster heads.

WANETs nodes can be equipped with different wireless media, according with devices and applications. The most used wireless standards are IEEE 802.11 [Standards association, 2012] and IEEE 802.15 [Standards association, 2005][Gislason, 2008], that transmit radio signals at different frequencies (from 2.4 to 5 GHz). Unlike traditional infrastructure-based wireless networks, nodes of WANET don't require IP address reconfiguration when they move in different network access points. WANETs include the following sub-categories: wireless sensor networks (WSNs), wireless mesh networks (WMNs), and vehicular networks (VANETs). WSNs consist in a collection of devices that can sense the environment and communicate informations through collector nodes, named sink. Sensors have limited computational capabilities and they are generally powered by an external battery with a limited capacity. WMNs provide interoperability among different networks and also a flexible access to Internet. They offer the possibility to connect heterogeneous devices using different wired and wireless media types. Moreover, WMNs offer important proprieties such as self-configuration, self-healing and self-management. VANETs were designed to improve the safety of vehicles which automatically warn nearby vehicles about movements averting them regarding dangerous situations.

2.1 Security attacks

WANETs can be openly accessible and adversaries can easy gain access to wireless communications without breaking any physical barriers. Hence, they result vulnerable to different security attacks [Di Pietro et al., 2014]. Table 2.1 reports the different type of attacks to the physical layer: *Jamming*, *Tampering*, *Eavesdropping*, and *Node Replication*. Jamming [Mpitziopoulos et al., 2009] is a denial of service (DoS) active attack which disrupts network communications. An adversary introduces noise in the carrier transmitting radio signals at target frequency or multiple frequencies. Jamming can be executed continually or at random intervals. A possible countermeasure is the adoption of spread spectrum technologies (FHSS or DSSS) [Pickholtz et al., 1982], that increase the resistance to radio interferences. Tampering is another example of active attack. An adversary attempts to steal data stored in nodes memory (such as cryptographic keys), with the intent of compromising their secrecy. Possible countermeasure are the design of tamper-resistant nodes or the implementation of defence mechanisms able to detect tampering attempts. Furthermore, a node could run a self-terminate procedure erasing cryptographic keys and all sensitive informations stored in memory. An adversary can easily perform an eavesdropping attack listening network communications. Thereafter, it can analyze captured data obtaining sensitive information such as events occurred, routing paths and commands. Data encryption can be an acceptable countermeasure to this last attack. Node Replication attack consists in a indefinitely replication of network nodes. To prevent node replication, one or more legitimate nodes should be used as traffic monitors to control network activities.

Attack	Target	Countermeasures
Jamming	Service integrity and availability	Spread Spectrum techniques, Jamming detection
Tampering	Service integrity and availability	Tamper-resistant hardware, Tamper-resistant software
Eavesdropping	Privacy and confidentiality	Data encryption
Node Replication	Data Integrity	Traffic monitors

Tab. 2.1: Attacks against the physical layer.

Table 2.2 reports different type of attacks to the link layer: *Collision*, *Exhaustion*, *Unfairness* and *Sleep Deprivation*. A collision occurs when a radio signal is sent at the same time and frequency of others signals. An adversary causes collisions in order to corrupt network messages and reduce node resources. A possible countermeasure is the adoption of error-detecting and error-correcting codes to recover corrupted messages. Moreover, an adversary could cause energy exhaustion by attempting continuous service requests. A networking rate limitation allows nodes to discard excessive requests. An unfairness attack consists in the transmission of a large number of messages, in order to make difficult for legitimate nodes the use of communication channel. This attack can be considered a denial of service (DoS), limited to a short time period. WANET nodes can remain in sleep mode for a long time preserving their residual power. The Sleep Deprivation attack aims to awake nodes with the deliberate intention of drain their power and reduce drastically their lifetime.

Attack	Target	Countermeasures
Sleep Deprivation	Service integrity and availability	Authentication
Unfairness	Service integrity and availability	Error-detecting, Error-correcting
Exhaustion	Privacy and confidentiality	Rate limitation
Collision	Data Integrity	Error-correcting

Tab. 2.2: Attacks against the link layer.

Table 2.3 reports the different type of attacks to network layer which aim to interruption of network availability. An adversary can perform a *Hello flooding* attack sending a large number of hello messages to its neighbors. In this way, it will quickly drain their residual power. Using message authentication is possible to mitigate hello flooding attacks.
The *Sink, Black and Wormhole attacks* aim to route network message through malicious nodes. In particular, sink and black hole attacks aim to drop all routing packets. Instead, in order to perform an wormhole attack, an adversary needs to compromise two or more nodes in different position of the network. The compromised nodes fake routing paths that appear shorter than original one. They confuse routing mechanisms which rely on metric distances among nodes.

Attack	Target	Countermeasures
Hello flooding	Network availability	Entity authentication
Synk, Wormhole, Black	Network availability	Entity authentication
Packets alteration, injection	Data integrity	Message authentication

Tab. 2.3: Attacks against Network Layer.

Concerning network layer, we distinguish *unicast*, *multicast* and *broadcast* communications. Unicast is a communication between a single sender and a single receiver. Indeed, If some node needs to send messages to multiple nodes, it will have to send multiple unicast messages, each message addressed to a specific node. Alternatively, multicast communications disseminate messages from a single sender to multiple receiver (e.g. a groups of nodes). A node can send messages to all network nodes using broadcast communications.

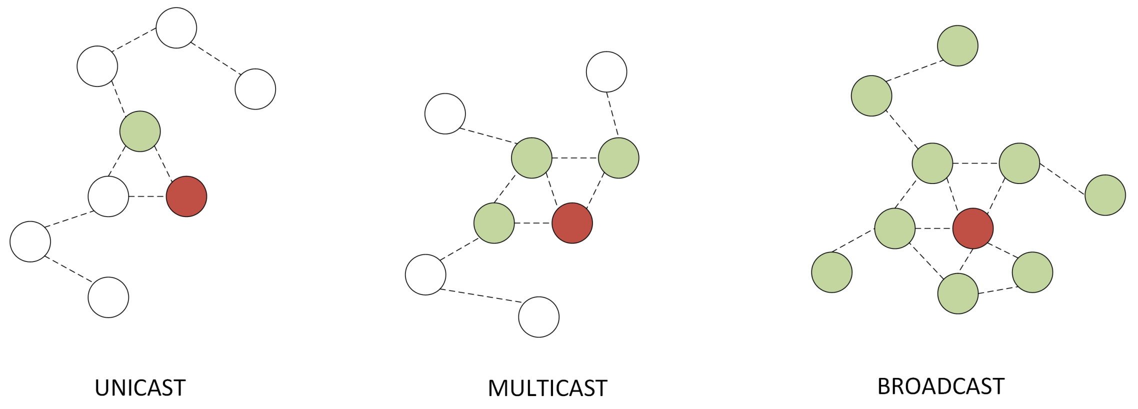


Fig. 2.2: Instance of unicast, multicast and broadcast communications. The red and green colors indicate sender and receiver nodes respectively.

2.2 Authentication mechanisms

Entity authentication is the process whereby one party is assured (through acquisition of corroborative evidence) of the identity of others parties involved in a protocol [Menezes et al., 1996]. Entity authentication is an essential security service to prevent attacks against WANETs, since cryptographic services like confidentiality, integrity and non-repudiation depend on authentication. A trust third party (TTP), can offer key distribution services deploying cryptographic materials on nodes through a secure communication channel. Nodes use cryptographic materials to carrying out authentication processes and then establish secure association with others through key establishment protocols.

2.2.1 Certificate-Based key distribution

We distinguish symmetric and asymmetric key distribution. Both require a TTP, which distributes secret keys preserving their confidentiality. If we assume a network of n nodes, a TTP can give rise to three possible symmetric key distribution schemes, as follows:

1. A unique key k for all nodes of the network. This approach is the simplest, but it offers lowest level of security, since compromise a single node could disrupt the entire network.

2. Each node has $n-1$ keys. This way, each pair of nodes shares a distinct secret key, for a total number of $n(n-1)/2$ keys. This approach results an unacceptable number of keys when the network size increases and thus can not be considered a scalable approach.

3. An equal-sized pool of keys from the set of all possible keys. In this scheme previously proposed by [Eschenauer and Gligor, 2002], each node has probability P of sharing a key with other nodes in the network. If two nodes don't share a key, they can do that through a chain of third nodes in the network.

A *certification authority (CA)* can be used for asymmetric key distributions. It releases digital certificates which contain a *data part* and a *signature part*. The data part includes node identity and the corresponding public key. The signature part consists in CA's signature over the data part, in order to bind univocally node identity to its public key.

Distribute certification authority (DCA) can offer more resilience and fault tolerance than a single CA. As shown in Table 2.4, DCA consists in a sub-group ($m < n$) of special nodes having t different partial signature keys (denoted as shares). A node's certificate is signed with a threshold mechanism collecting t shares. A node combiner collects the t partial signatures and obtains a full signature. The goal of this approach is a trade-off between mitigation of impersonation attacks and fault tolerance, at the cost of more resources and network bandwidth. In the fully DCA (FDCA) approach, all network nodes are DCA nodes ($m = n$). It allows for a better fault tolerance and higher security than DCA, at the cost of a more expensive setup.

Approach	Share	Sever nodes	Fault tolerance
CA	-	1	*Low*
DCA	$t <= m$	$m < n$	*Medium*
FDCA	$t <= m$	$m = n$	*High*

Tab. 2.4: Asymmetric certificate-based key distribution schemes.

2.2.2 ID-based key distribution

ID-based key distribution relies on a *key generation center (KGC)*. After that the KGC generates its master secret key and a set of system public parameters, each node can be registered to KGC obtaining a private key associated to its identity. An ID-based key distribution offers different advantages, due to the fact that identities can be free-text strings suitably chosen to convey appropriate information. It requires each node storing in memory only its private keys and the public parameters. This can be much less bandwidth and memory consuming than certificate management in CA's schemes. A possible drawback of identity-based cryptosystems is that they are exposed to the key escrow problem [Lee et al., 2004], since the KGC knows by construction the secret keys of all its users. That is clearly unacceptable in open networks such as the Internet. However, this is not a concern in the application scenario we are interested in, since WANETs are typically deployed by an off-line KGC.

2.3 Key establishment

Accordingly to paper of [Menezes et al., 1996], a *key establishment protocol* provides a shared secret key between two or more parties for encryption, message authentication and others cryptographic purposes. Formally, a *secure key establishment* protocol is said to have the *implicit key authentication* property if unauthorized parties can't gain access to the shared secret key. Furthermore, the *key confirmation* property means that any party has evidence of the possession of the shared secret key by the others parties involved in the protocol. A protocol with both proprieties is said to have *explicit key authentication*.
An output of a key establishment protocol is typically a *session key* valid only for a short time period (usually a communication session). The use of session keys can provide different advantages, such as:

- Limited exposure of data in the event of session key compromise.
- Limited number of ciphertexts available for cryptanalysis.
- Short-term storage of possible shared keys.

A key establishment protocol is said to have *perfect forward secrecy* if a session key is not compromised in case one of the (long-term) private keys will be compromised in the future. Instead, the protocol is said to have *key independence*, if knowledge of some session keys doesn't allow an adversary to compromise past or future session keys.
Key establishment protocols can be subdivided into *key transport* and *key agreement* protocols. They are defined by [Menezes et al., 1996] as follows:

Definition 2.3.1. *A key transport protocol is a key establishment technique where one party creates or otherwise obtains a secret value, and securely transfers it to the other(s).*

Definition 2.3.2. *A key agreement protocol or mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these, (ideally) in such a way that no party can predetermine the resulting value (full contributory key agreement).*

Key agreement protocols are thus contributory mechanisms whereas the parties cooperates in the same way to establish a common secret. The number of protocol participants can range from two (two-party key agreement) to group of parties (group key agreement or a.k.a conference keying, although this last is a somewhat obsolete term by now). In the next Section, we will illustrate the first two-party key agreement protocol published by [Diffie and Hellman, 1976], that provides the basis for a variety of key agreement protocols.

2.4 Dieffie-Hellman protocol

In 1976 in their seminal academic work considered the startup of public key cryptography, [Diffie and Hellman, 1976] (DH) presented a key agreement protocol that allows two parties to obtain a shared secret key over a public communication channel. Given a large prime number l and the multiplicative group $\mathbb{Z}_l^*$ with generator g, Alice a Bob execute the protocol as follows:
Alice chooses a random number $x \in \mathbb{Z}_l^*$ and Bob chooses a random number $y \in \mathbb{Z}_l^*$. Alice sends g^x to Bob and Bob sends g^y to Alice. Alice calculates $(g^x)^y$ and Bob computes $(g^y)^x$ so that $K = (g^x)^y = (g^y)^x = g^{xy}$

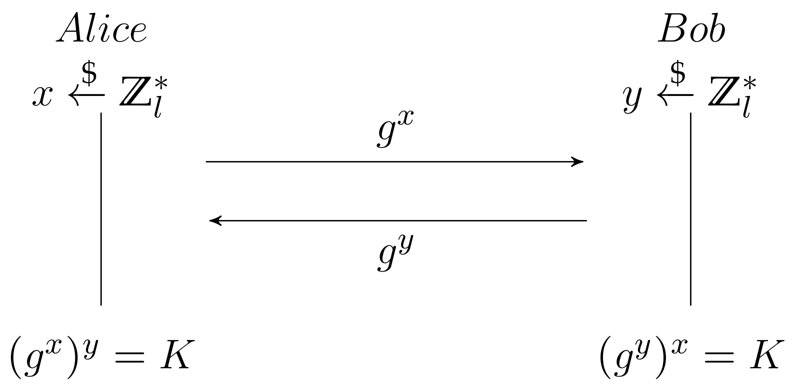

Alice
$x \xleftarrow{\$} \mathbb{Z}_l^*$
Bob
$y \xleftarrow{\$} \mathbb{Z}_l^*$
g^x
g^y
$(g^x)^y = K$
$(g^y)^x = K$

Fig. 2.3: DH protocol.

The DH key agreement protocol is secure against passive adversaries, assuming that an Eavesdropper E who sees g^x and g^y is not able to calculates g^{xy}. An adversary could perform different kind of security attacks against DH protocol. For example, it could impersonate the parties or prevent from carrying out the protocol session. Moreover, it could delete messages exchanged overcoming communication, causing a denial of service (DoS). Referring to secure key establishment properties described in the previous Section, the DH protocol is also vulnerable to *know-key attacks* [Menezes et al., 1996]. The involved parties can preserve key independence using different secret keys in protocol runs.

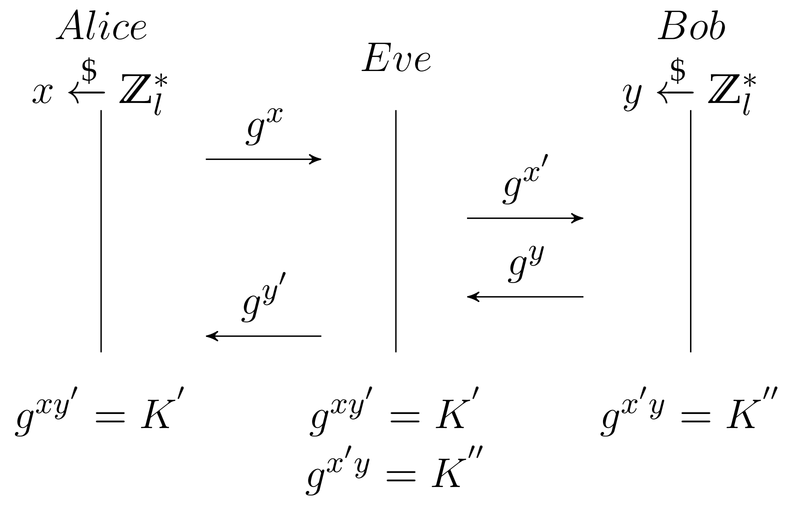


Fig. 2.4: Man in the middle attack against DH protocol.

Figure 2.4 illustrates a man in the middle attack [Desmedt, 2011]. In this case, Eve intercepts g^x and g^y, replacing them with $g^{x'}$ and $g^{y'}$. Alice will believe that its shared key is $g^{xy'}$ and Bob will believe that its shared key is $g^{x'y}$. Both values are available to Eve, who can impersonate Bob with Alice and Alice with Bob, thus eavesdropping their confidential communications.
Figure 2.5 illustrates the station-to-station (STS) protocol of [Diffie et al., 1992] which is one of authenticated versions of DH protocol. It makes use of public key certificates (PKCs) and symmetric encryption, providing implicit key authentication and perfect forward secrecy.
Also in this case, Alice chooses a random number $x \in \mathbb{Z}_l^*$ and sends g^x to Bob. Bob chooses a random number $y \in \mathbb{Z}_l^*$, computes the secret key $k = g^{xy}$ and encrypts with key k its signature over the concatenation of exponentials (g^y, g^x). Bob sends to Alice the message g^y, $Cert_B$, $E_k(S_B(g^y, g^x))$. Alice also computes the key $k = g^{xy}$, decrypts and verifies Bob's signature using asymmetric public key included in the his digital certificate. After that, Alice encrypts its signature over the concatenation of exponentials (g^y, g^x) and sends to Bob the message $Cert_A$, $E_k(S_A(g^y, g^x))$. Bob decrypts and verifies Alice's signature using asymmetric public key included in her digital certificate. Thus, they obtain a mutual authentication and the shared secret key K at same time.

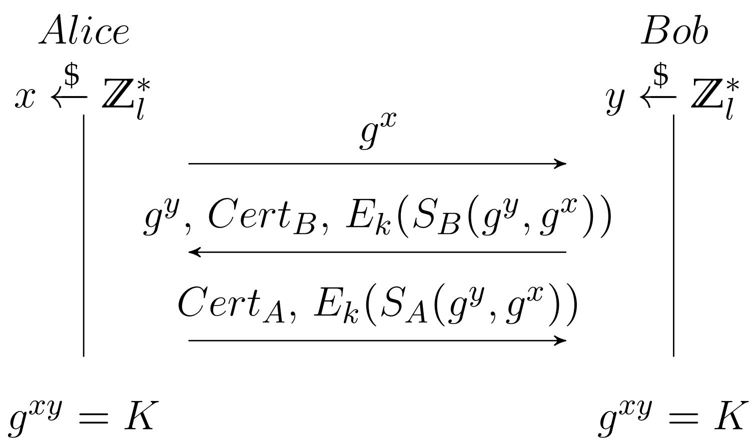

$Alice$
Bob
$x \xleftarrow{\$} \mathbb{Z}_l^*$
$y \xleftarrow{\$} \mathbb{Z}_l^*$
g^x
$g^y, Cert_B, E_k(S_B(g^y, g^x))$
$Cert_A, E_k(S_A(g^y, g^x))$
$g^{xy} = K$
$g^{xy} = K$

Fig. 2.5: STS protocol.

Theoretical background

In this Chapter we briefly review some basic facts about discrete logarithms, elliptic curves and bilinear pairings, that notions and/or assumptions will be used in the sequel. More details about these topics have been previously published by [Koblitz, 1994] and [Silverman, 2009].

3.1 Finite fields and discrete logarithm systems

If $\mathbb{F}_q = \mathbb{F}_q(+,\cdot)$ denotes a field with q elements, then it follows that $q = p^d$, where p is a prime number and d is a positive integer. Moreover, for every prime power $q = p^d$ there exists a field $\mathbb{F}_q$ of q elements (up to isomorphism). The sum of p times the multiplicative identity 1 equals the additive identity 0 and it is called the *characteristic* of $\mathbb{F}_q$. The integer d is the *extension degree* of the field, since $\mathbb{F}_q$ contains the prime field $\mathbb{F}_p$ and it is a vector space of dimension d over $\mathbb{F}_p$. The $(q-1)$ non zero elements of $\mathbb{F}_q$, form an abelian group with respect to multiplication and it is often denoted as $\mathbb{F}_q^* = \mathbb{F}_q^*(\cdot)$.

Each point $P \in \mathbb{F}_q^*$ admits a least positive integer l such that $P^l = 1$, which is called the *order* of P and divides $(q-1)$. Moreover, if $\varphi(n)$ denotes the number of positive integers $i < n$ such that $\gcd(i,n) = 1$, there are exactly $\varphi(q-1)$ elements $G \in \mathbb{F}_q^*$ having order $l = q-1$. Each such G is called a *generator* of $\mathbb{F}_q$, since $\langle G \rangle = \{G^i : \ i = 1, \dots q-1\}$ coincides with $\mathbb{F}_q^*$. We will suppose from now to choose $P \in \mathbb{F}_q^*$ in such way that its order l is prime, so that l is a prime factor of $q-1$ and $\gcd(l,q) = 1$. The cyclic multiplicative group generated by any of such

P, $\langle P\rangle = \{P^i : \ i = 1, \dots, l\}$, is a subgroup of $\mathbb{F}_q^*$ which has order l and exhausts all no zero residue classes modulo l. Thus, we can consider the following problem in $\langle P\rangle$:

Definition 3.1.1 (Discrete Logarithm problem)**.** *Let $P \in \mathbb{F}_q^*$ be of prime order l. The* discrete logarithm *(DL) problem in $\mathbb{G} = \langle P\rangle$ is as follows: given $Q \in \mathbb{G}$, find $x \in \mathbb{Z}_l$ such that $Q = P^x$. The integer x (which is unique modulo l) is called the discrete logarithm of Q to the base of P, and denoted* $\log_P Q$*.*

The size q of the finite field and that of the *cofactor* $c = (q-1)/l$ are obviously related to the difficulty of the DL problem in $\langle P\rangle$. The standard rule is to choose q to be a very big integer and the cofactor c to be as small as possible.[1] The two best known algorithms to solve the DL Problem for general finite fields $\mathbb{F}_q$ are both based on the *index calculus method* [Diem, 2011]. The first algorithms is derived from [Schirokauer, 2000], and if $d < \sqrt{\ln p}$, attains a sub-exponential expected running time of

$$\exp((1.923 + o(1))(\ln q)^{1/3}(\ln\ln q)^{2/3} \tag{3.1}$$

The second is due to [Coppersmith, 1984], applies only to $d > (\ln p)^2$, and runs substantially faster than [Schirokauer, 2000], but only on fields of small characteristic. For example, if $p = 2$, Coppersmith's algorithm expected running time is given by equation (3.1) with factor 1.923 replaced by 1.588. The DL problem is at least as difficult as the Diffie-Hellman problems in $\mathbb{G}$, that are at the basis of many cryptosystems.

Definition 3.1.2 (Diffie-Hellman problems)**.** *Let $P \in \mathbb{F}_q^*$ be of prime order l. The* computational (CDH) *and* decisional (DDH) *Diffie-Hellman problems in $\mathbb{G} = \langle P\rangle$ are as follows:*

[1] it is suggested this results in a prime l at least of order 2^{1024}, and this threshold is going to increase over time.

1. CDH*: given* $Q = P^x$ *and* $R = P^y$ *find* P^{xy}*, where* x, y *are randomly chosen in* $\mathbb{Z}_l$*.*

2. DDH*: given* $Q = P^x$*,* $R = P^y$ *and* $S = P^z$ *decide if* (Q, R, S) *is a DH-triple, i.e. if* $z = xy$*, where* x, y, z *are randomly chosen in* $\mathbb{Z}_l$*.*

3.2 Elliptic curves

An *Elliptic curve* over $\mathbb{F}_q$, denoted as $E(\mathbb{F}_q)$, is the set of solutions (x, y) of an equation in $\mathbb{F}_q$ of the form:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{3.2}$$

together with an additional point at infinity O, which represents the direction of the y-axis. Equation (3.2) reduces to a simpler form according to the field characteristic p, as follows:

$$\begin{aligned}
y^2 + xy &= x^3 + ax + b && (p = 2,\ E \text{ supersingular})\\
y^2 + xy &= x^3 + ax^2 + b && (p = 2,\ E \text{ ordinary})\\
y^2 &= x^3 + ax^2 + bx + c && (p = 3)\\
y^2 &= x^3 + ax + b && (p > 3)
\end{aligned}$$

The number of points of an elliptic curve $E(\mathbb{F}_q)$, denoted $|E(\mathbb{F}_q)|$, is called the *order* of the curve and is given by $|E(\mathbb{F}_q)| = q + 1 - t$, where $|t| \leq 2\sqrt{q}$. If $q = p^d$ $(d > 1)$, then for each $|t| \leq 2\sqrt{q}$ such that p does not divide t there exists an elliptic curve defined over $\mathbb{F}_q$ with $|E(\mathbb{F}_q)| = q + 1 - t$. If $q = p$, then for each $|t| < 2\sqrt{p}$ there exists an elliptic curve defined over $\mathbb{F}_p$ with $|E(\mathbb{F}_p)| = p + 1 - t$. In particular, for any field $\mathbb{F}_p$ of prime order p, all *supersingular* elliptic curves over $\mathbb{F}_p$ have exactly $p + 1$ elements [Silverman, 2009]. From now, given an arbitrary curve

over a finite field $E(\mathbb{F}_q)$, we will denote with "+" also the binary operator which to $P, Q \in E(\mathbb{F}_q)$ associates the point on $E(\mathbb{F}_q)$ which is symmetric w.r.t. the x-axis to the intersection of the straight line passing through P and Q (or tangent to E, if $P = Q$). Indeed, likewise the set of integers with the standard operation of addition between numbers, it can be shown that $E(\mathbb{F}_q)$ has an abelian group structure with respect to the previous operation, called *addition of points*. Since the straight line "passing through" P and O is the line from P having the direction of the y-axis, it should be clear that $P + O = O + P = P$ for each $P \in E(\mathbb{F}_q)$; that is, the point at infinity O represents the identity element of the group.
Groups over elliptic curves are particularly relevant in cryptography, since there are well known curves for which the DL problem (and the Diffie-Hellman problems) are much more difficult than in standard groups. Indeed, the best generic algorithm known for solving the DL problem over elliptic curves is the *rho method* of [Pollard, 1978], which has an expected running time of order $\sqrt{|E(\mathbb{F}_q)|}$, that is exponential in $\ln q$.

3.3 Bilinear pairings and co-Diffie-Hellman problems

Some special mappings between groups, called *bilinear pairings*, were firstly described by [Menezes et al., 1993] that reported the DL problem over elliptic curves to that over a standard group in a field; this way, the hardness of the native DL problem can be reduced, provided that the co-domain of the mapping is of suitable size. A notable set of cryptanalytic methods work on this idea, alongside with strategies for choosing curves in order to circumvent related attacks. Later on, bilinear pairings have also been used to realize cryptosystems with improved efficiency and/or special properties, like the signature schemes discussed in this thesis.

Definition 3.3.1 (Bilinear Pairing)**.** *Let $\mathbb{G}$, $\mathbb{G}'$ be two (additively written) cyclic* *groups of prime order l, and let $\bar{\mathbb{G}}$ be a multiplicative group. The mapping:*

$$e : \mathbb{G} \times \mathbb{G}' \to \bar{\mathbb{G}}$$

is said a bilinear pairing *if it has the following properties:*

1. *Bilinearity: for all $(P, P') \in \mathbb{G} \times \mathbb{G}'$ and $a, b \in \mathbb{Z}_p^*$ then $e(aP, bP') = e(P, P')^{ab} \in \bar{\mathbb{G}}$.*

2. *Not-degeneracy: for all $(P, P') \in \mathbb{G} \times \mathbb{G}'$ then $e(P, P') \neq 1$.*

3. *Computability: there is an efficient algorithm which computes $e(P, P')$ for all $(P, P') \in \mathbb{G} \times \mathbb{G}'$.*

With this setup we obtain natural generalizations of the CDH and DDH problems, that reduce to standard CDH and DDH when $\mathbb{G} = \mathbb{G}'$:

Definition 3.3.2 (co-Diffie-Hellman problems)**.** *Let $\mathbb{G} = \langle P \rangle$, $\mathbb{G}' = \langle P' \rangle$ be two (additively written) cyclic groups of prime order l. The* computational (co-CDH) *and* decisional (co-DDH) *co-Diffie-Hellman problems on $(\mathbb{G}, \mathbb{G}')$ are as follows:*

1. co-CDH*: given $Q' = xP'$ and $R = yP$ find xyP, where x, y are randomly chosen in $\mathbb{Z}_l$.*

2. co-DDH*: given $Q' = xP'$, $R = yP$ and $S = zP$ decide if (Q', R, S) is a co-DH-triple, i.e. if $z = xy$, where x, y, z are randomly chosen in $\mathbb{Z}_l$.*

Hardness of the co-CDH problem on $(\mathbb{G}, \mathbb{G}')$ implies the hardness of the DL problem on $\mathbb{G}'$, but in general no more. In other words, the DH problems could be easy to solve in $\mathbb{G}'$, and the DL problem could be easy in $\mathbb{G}$ (and thus the DH problems, too). Analogously, and with a subtle difference w.r.t. the standard DDH problem, the hardness of the co-DDH problem on $(\mathbb{G}, \mathbb{G}')$ implies only that of the DL problem on $\mathbb{G}'$.

Things change radically if one suppose that a bilinear pairing e and an efficiently computable isomorphism ψ exist such that:

$$e : \mathbb{G} \times \mathbb{G}' \to \bar{\mathbb{G}}, \quad \psi : \mathbb{G}' \to \mathbb{G}$$

Indeed, it can be proved that:

Proposition 3.3.1. *Let e be a bilinear pairing as in definition (3.3.1). Let ψ be a bijection from $\mathbb{G}'$ in $\mathbb{G}$ such that $\psi(Q' + R') = \psi(Q') + \psi(R')$. Then:*

- *The co-CDH problem on $\mathbb{G} \times \mathbb{G}'$ is no more difficult than each one of the DL problems on $\mathbb{G}$, $\mathbb{G}'$ and on the ground field in which $\bar{\mathbb{G}}$ is constructed.*
- *The co-DDH problem on $\mathbb{G} \times \mathbb{G}'$ can be efficiently solved.*

This *gap* between the two Diffie-Hellman problems is at the basis of many modern cryptosystems, as the short signature schemes introduced in paper of [Boneh et al., 2004]. We wish to stress that, as it should be clear from what stated before, the existence of the isomorphism ψ is *mandatory* for the security of these cryptosystems. This paper gives an example for which the co-CDH problem on $(\mathbb{G}, \mathbb{G}')$ is believed to be hard and yet their signature scheme is insecure.

3.3.1 Bilinear pairings from the tate pairings

An important class of bilinear pairings for elliptic curves over finite fields can be deduced from the pairing [Frey et al., 1999]. These pairings maps couples of points placed in two cyclic subgroups $\mathbb{G}$, $\mathbb{G}'$ of prime order l in $E(\mathbb{F}_q)(+)$ into an order l subgroup $\bar{\mathbb{G}}$ of some extension field $\mathbb{F}_{q^k}$. The degree $k = k(l)$ of this extension is called the *embedding degree* of the pairing (for the order l), and it is the smallest positive integer k such that l divides $q^k - 1$.
More precisely, let $P \in E(\mathbb{F}_q)(+)$ be a point of prime order l, and suppose $\gcd(l, q - 1) = 1$ (i.e. $k > 1$). If E is supersingular, then E admits an endomorphism

$\phi : E \to E$ for which $\phi(P) \notin \langle P \rangle$, which is called a *distortion map*. In this case, it can be shown that the mapping defined by:

$$e : (Q, R) \in \langle P \rangle \times \langle P \rangle \longrightarrow e(Q, R) = \tau(Q, \phi(R)) \tag{3.3}$$

where τ is the value assumed by the Tate pairing, is a symmetric (i.e. $\mathbb{G} = \mathbb{G}'$) bilinear pairing in the sense of definition (3.3.1). If E is ordinary, then no distortion maps exist. However, selecting $P' \notin \langle P \rangle$ results in the asymmetric bilinear pairing given by:

$$e : (Q, R') \in \langle P \rangle \times \langle P' \rangle \longrightarrow e(Q, R') = \tau(Q, R') \tag{3.4}$$

From Proposition 3.3.1, it follows that for a cryptosystem based on the previous setup the integers q, l and k must all be considered security parameters, and that they should satisfy the following conditions:

1. l should be sufficiently large so that known methods (i.e. exponential-time methods) for computing the DL problem in an order l subgroup of $E(\mathbb{F}_q)(+)$ are infeasible.

2. k should be sufficiently large so that index-calculus methods for solving the DL problem in $\mathbb{F}_{q^k}^*$ are infeasible.

3. q and k should be small enough so that arithmetic in $\mathbb{F}_q$ and $\mathbb{F}_{q^k}$ can be efficiently performed.

Table 3.1 shows values of the parameters q, l, k such that the security of DL-based cryptosystems related to the above setting of bilinear pairings is equivalent to that of a state-of-the-art symmetric-key cipher (e.g. AES) for common key lengths.

Security Level	q	l	k
80-bit key	2^{160}	2^{160}	7
80-bit key	2^{256}	2^{160}	4
80-bit key	2^{512}	2^{160}	2
128-bit key	2^{256}	2^{256}	12
128-bit key	2^{512}	2^{256}	6
128-bit key	2^{1024}	2^{256}	3
192-bit key	2^{384}	2^{384}	20
192-bit key	2^{512}	2^{384}	15
192-bit key	2^{1024}	2^{384}	8

Tab. 3.1: Security Levels of a DL-based system for different values of the size q of the ground field, the order l of the cyclic groups where the DL-system is defined, and the embedding degree k of the bilinear pairing.

As we have seen in the previous subsections, solvers of the DL problem in standard groups and in elliptic curve groups have sub-exponential and exponential expected running time, respectively. Then, in order to avoid attacks, the size q^k of the embedding field must be proportionally greater than the order l of the groups over elliptic curves as l increases. This is reflected by the behaviour of the embedding degree k versus the security level in case of a cofactor $c = 1$, as shown in Table 3.1.

3.4 ID-based cryptography and hashing function algorithms

The concept of ID-based signature (IBS) was introduced in the seminal paper of [Shamir, 1985]. The innovation was the use of identity attributes instead of public keys for data signature, thus avoiding the generation and management of users' public key certificates. That can significantly reduce the computational overheads introduced by cryptography for application security.

In the 2001, [Boneh and Franklin, 2001] solve the problem of ID-based encryption (IBE). In the years that followed, inspired to works of Shamir and Boneh, many IBE [Sakai and Kasahara, 2003],[Cocks, 2001],[Lynn, 2002] and IBS [Hess, 2003],[Barreto et al., 2005] were proposed.

An *identity-based signature (IBS) scheme* is a quadruple of probabilistic polynomial time (PPT) algorithms *IBS = (Setup, KeyGen, Sign, Vrf).*

- *Setup* takes as input a security parameter k and returns system parameters *params*. They include the system master-key s, the corresponding public key V and a description of a message space $\mathcal{M}$.

- *KeyGen* takes as inputs the master key s and a string $id \in \{0,1\}^*$, and returns a secret signature key $s(id)$.

- *Sign* takes as inputs the secret signature key $s(id)$ and a message $m \in \mathcal{M}$, and returns the signature Σ of m.

- *Vrf* takes as inputs V, id, m and Σ. It returns 1 if Σ is a valid signature of m related to id, and 0 otherwise.

The Setup and KeyGen algorithms are performed by a KGC. We also refer to these procedures user *Enrollment*. We remark that users can obtain more then one private signature keys $(s_1(id), s_2(id), .., s_n(id))$ from KGC.
In IBS schemes it is customary to shrink user identities id and messages m (which are binary strings of variable length) to elements in suitable groups. From a theoretical point of view, these shrinking mappings are often modeled as *random oracles (RO)*, i.e. functions whose outputs appear to an observer like uniformly distributed random variables. In practice, such functions are actually realized through cryptographic hash functions, as suggested in [Bellare and Rogaway, 1993]. For the IBS schemes provided in our implementation, we had to consider the two hash function functions.

$$h : \{0,1\}^* \to \mathbb{Z}_l^*, \quad H : \{0,1\}^* \to \mathbb{G} \ ,$$

where $\mathbb{G}$ is the group of prime order l considered in Section 3.3.1. The first function can be easily obtained by computing an hash function and keeping the most significant $\lceil \log_2 l \rceil$ bits of its digest, and finally by considering the integers that have these bits as their binary representation and fall in $\mathbb{Z}_l^*$. Indeed, it can be shown that truncating the output of a pseudo-random function to an arbitrary number of bits results again in a pseudo-random function. On the other hand, the SHA-2 family of hash function is supposed to emulate fairly well pseudo-random functions, and digest bit sizes up to 512 allow to encompass all the values of l required for elliptic curve cryptography. Our implementation makes use of the family member having 512 bit output, known as SHA-512. This function is considered secure as far as input messages are no more than $2^{128} - 1$ bits, a value far beyond the requirements of our applications.

The construction of function H is more involved. Since hashing directly onto a subgroup of an elliptic curve is a difficult task, we followed the construction for H given in [Boneh et al., 2004], which consists in what follows. Let suppose $p \geq 3$, then from equation (3.2) it follows that elliptic curves over $\mathbb{F}_q$ are defined by an equation of type $y^2 = f(x)$, where $f(x)$ is a quadratic polynomial. Let $E(\mathbb{F}_q)$ be a curve of order m, and let $P \in E(\mathbb{F}_q)$ be a point of prime order l, where l^2 does not divide m. Suppose now we have the function $H' : \{0,1\}^* \rightarrow \mathbb{F}_q \times \{0,1\}$, which likewise h can be built from a standard hash function. The following algorithm uses H' to hash strings in $\{0,1\}^*$ onto $\mathbb{G} = \langle P \rangle$ with a failure probability no more than $\delta > 0$, where δ can be chosen arbitrarily small.

MapToGroup

Given the function H' and $I = \lceil \log_2 \log_2 \delta^{-1} \rceil$, the Map to group algorithm does the following:

1. Set $i = 0$, where i is represented as an I-bit string.
2. Set $(x, b) = H'(i \| str)$, where $str \in \{0,1\}^*$.
3. If $f(x)$ is a quadratic residue in $\mathbb{F}_q$ then do:
   (a) Use b to choose between the two square roots $y_1 > y_0$ of $f(x)$ and set $P(str) = (m/p)(x, y_b) \in \mathbb{G}$.
   (b) If $P(str) \neq O$ then output $MapToGroup(str) = P(str)$ and stop; otherwise, continue with step 4.
4. Increment i and go to step 2, if $i = 2^I$ report failure.

3.5 Point compression

If the ground field $\mathbb{F}_q$ $(q = p^d)$ is of odd characteristic p, then all the IBS schemes discussed later can take advantage of *point compression* [Boneh et al., 2004],[Zhang et al., 2004]. It is because of compression that conventional pairing-based schemes gets half-sized signatures compared to ECDSA. Indeed, since for an elliptic curve there are at most two points with the same x-coordinate of equation (3.2) for $p \geq 3$, then one can save such coordinate and one bit to uniquely identify a point in $\mathbb{G}$ (or $\mathbb{G}'$).

ID-based signature

In this Chapter, we present two new ID-based signature schemes, named *IBS-1* and *IBS-2*. They make use of bilinear pairing to get shorter signatures and can be considered the ID-based counterparts of [Boneh et al., 2004] (BLS) and [Zhang et al., 2004] (ZSS) conventional signature schemes. The new signature schemes will be compared with the signature scheme of [Gentry and Silverberg, 2002] (GS-IBS). The users obtain signatures keys making an *Enrollment* to KGC which runs Setup and KeyGen algorithms (see detail in Section 3.4). The first task of the Setup algorithm is to define the elliptic curve $E(\mathbb{F}_q)$ to be used, and a point $P \in E(\mathbb{F}_q)$ of prime order l. Moreover, this algorithm computes the description of a suitable Tate-based pairing e, plus that of its related distortion map ϕ or isomorphism ψ, depending if the elliptic curve is supersingular or ordinary, respectively. The *KeyGen* computes two kinds of signature keys, since the IBS-1 and the GS-IBS schemes adopt the same type. It's important to notice that users signature keys are now points in $\mathbb{G}(+) = \langle P \rangle$, whereas they are integers in $\mathbb{Z}_l^*$ in the related conventional signature schemes BLS and ZSS.

Setup Algorithm

The scope of the Setup algorithm is the setting of public parameters for the intended service. The KGC does the following:

- Choice of an elliptic curve $E(\mathbb{F}_q)$, and a point $P \in E$ of prime order l with $\gcd(l, q-1) = 1$.
- If E is supersingular then:
  - Selection of a distortion map ϕ in E with $P' = \phi(P)$, and definition of the symmetric bilinear pairing e given by equation (3.3).

  Otherwise, if E is ordinary, then:
  - Selection of a point $P' \in E(\mathbb{F}_{q^k})$ of order l which is linearly independent of P, and definition of the asymmetric bilinear pairing e given by equation (3.4).
- Choice of a pseudo-random integer $s \in \mathbb{Z}_l^*$ as its master-key, and computation of its public verification key $V' = sP'$.
- Publication of E, e, l, P, V' and of the two hash functions:

$$H: \{0,1\}^* \to \mathbb{G}\ , \quad h: \{0,1\}^* \to Z_l^*$$

KeyGen Algorithm

The scope of the KeyGen algorithm is the generation of private signature keys for a set of users registered through suitable identities. The KGC does the following:

- For each binary string id in a suitable finite set:
- Computation of digests $H(id)$ and $h(id)$.
- Computation of the elements of $\mathbb{G}(+) = \langle P \rangle$:

$$S_1(id) = s\,H(id), \quad S_2(id) = \frac{P}{s + h(id)} \tag{4.1}$$

which represent private signature keys corresponding to the identity id.

4.1 IBS-1 and IBS-2 schemes

The IBS-1 scheme, illustrated in Figure 4.1, uses the signature key $S_1(id)$ returned by algorithm KeyGen. The signer id uses its private signature key $S_1(id)$ given by expression (4.1) and the public parameters returned by the Setup algorithm in order to compute the signature $(\Sigma, R') \in \mathbb{G} \times \mathbb{G}'$ of message m. The verifier can check the signature due to the knowledge of the signer identifier id and the public parameters returned by the Setup algorithm.

IBS-1 signature scheme

- **Sign**: the signer id does the following
  - Choice of a pseudo-random integer $r \in \mathbb{Z}_l^*$, and computation of the public element $R' = rP'$.
  - Computation of the element of $\mathbb{G}(+) = \langle P \rangle$:

$$\Sigma = \frac{S_1(id)}{r + h(m)} \tag{4.2}$$

  The couple (Σ, R') represents the signature by user id for message m.

- **Vrf**: the verifier does the following
  - Computation of digests $H(id)$, $h(m)$ and point $R' + h(m)P'$.
  - Check of the equality

$$e(\Sigma, R' + h(m)P') = e(H(id), V') \tag{4.3}$$

  and acceptance of the signature only if expression (4.3) is true.

It is easy to verify the correctness of the IBS-1 signature scheme. Substituting the value $S_1(id)$ given by (4.1) in the expression (4.2) of Σ, from the Bilinearity of e.

Proposition 4.1.1. *Suppose that no adversaries interfere with execution of signature scheme. For equation (4.3), the scheme is correct:*

$$\begin{aligned}e(\Sigma, R' + h(m)P') &= e\left(\frac{sH(id)}{r + h(m)}, (r + h(m))P'\right)\\ &= e(sH(id), P')\\ &= e(H(id), V')\ .\end{aligned}$$

Proposition 4.1.2. *Let $\mathbb{G} = \langle P \rangle$ and $\mathbb{G}' = \langle P' \rangle$, as returned by algorithm Setup, constitute a group pair of prime order l for which the co-CDH problem is hard to solve. Let the two functions $H : \{0,1\}^* \to \mathbb{G}$ and $h : \{0,1\}^* \to \mathbb{Z}_l^*$, returned by algorithm Setup, behave as random oracles. Then the IBS-1 scheme is secure against existential forgery under adaptive chosen-message attacks.*

Proof. Under the given assumptions both the BLS and ZSS schemes are secure against existential forgery under adaptive chosen-message attacks, as respectively proved in [Boneh et al., 2004] and [Zhang et al., 2004]. The existential unforgeability [Goldwasser and Micali, 1982] of BLS implies that it is unfeasible to compute a forged signature key $\bar{S}_1(id) \neq S_1(id)$ for any given id and any given master verification key V. On the other hand, the existential unforgeability of ZSS implies that it is unfeasible to compute a forged signature $\bar{\Sigma} \neq \Sigma$ for any given message m and any given public element R'. □

Scheme IBS-2 requires just one pairing computation for signature verification, instead of the two ones required by the IBS-1 scheme. This is because, in analogy with the ZSS scheme, one pairing computation can be precomputed by the KGC. This scheme, described by Figure 4.1, turns out by applying the ZSS signature $S_2(id)$ returned by algorithm KeyGen for id, and then using such signature as the base point to compute again a ZSS signature for message m. The signer id uses its private signature key $S_2(id)$ given by (4.1) and the public parameters returned by the Setup algorithm in order to compute the signature $(\Sigma, Q) \in \mathbb{G} \times \mathbb{G}$ for message

m. The verifier can check the signature thanks to the knowledge of the signer identifier id and the public parameters returned by the Setup algorithm.

IBS-2 signature scheme

- **Sign**: the signer id does the following
  - Choice of a pseudo-random integer $r \in \mathbb{Z}_l^*$.
  - Computation of the element of $\mathbb{G}(+) = \langle P \rangle$

$$\Sigma = \frac{S_2(id)}{r + h(m)} \tag{4.4}$$

  - Computation of the public element $Q = r\Sigma$.

  The couple (Σ, Q) represents the signature by user id for message m.

- **Vrf**: the verifier does the following
  1. Computation of digests $h(id)$, $h(m)$ and the related points $Q + h(m)\Sigma$, $V' + h(id)P'$
  2. Check of the equality

$$e(Q + h(m)\Sigma, V' + h(id)P') = e(P, P') \tag{4.5}$$

  and acceptance of the signature only if (4.5) is true.

The correctness of the IBS-2 scheme follows from the bilinearity of e and by substituting the expression of $S_2(id)$ in (4.4):

Proposition 4.1.3. *Suppose that no adversaries interfere with execution of signature scheme. For equation (4.5), the scheme is correct:*

$$\begin{aligned} e(P, P') &= e\left(P, \frac{s+h(id)}{s+h(id)} \frac{r+h(m)}{r+h(m)} P'\right) \\ &= e\left(\frac{r+h(m)}{(r+h(m))(s+h(id))} P, V' + h(id)P'\right) \\ &= e(Q + h(m)\Sigma, V' + h(id)P') \ . \end{aligned}$$

Proposition 4.1.4. *Let $\mathbb{G} = \langle P \rangle$ and $\mathbb{G}' = \langle P' \rangle$, as returned by algorithm Setup, constitute a group pair of prime order l for which the co-CDH problem is hard to solve. Let the function $h : \{0,1\}^* \to Z_l^*$, returned by algorithm Setup, behave as a random oracle. Then scheme IBS-2 is secure against existential forgery under adaptive chosen-message attacks.*

Proof. Under the given assumptions the ZSS scheme is secure against existential forgery under adaptive chosen-message attacks [Zhang et al., 2004]. On the other hand, the existential unforgeability of ZSS implies that: (a) it is unfeasible to compute a forged signature key $\bar{S}_2(id) \neq S_2(id)$ for any given id and any given master verification key V; (b) it is unfeasible to compute a forged signature $\bar{\Sigma} \neq \Sigma$ for any given message m and any given public element Q. That concludes the proof. □

In the signature scheme GS-IBS, the signer id uses its private signature key $S_1(id)$ given by (4.1) and the public parameters returned by the Setup algorithm in order to compute the signature $(\Sigma, R') \in \mathbb{G} \times \mathbb{G}'$ for message m. The verifier can check the signature due to the knowledge of the signer identifier id and the public parameters returned by the Setup algorithm.

4.2 GS-IBS scheme

GS-IBS signature scheme

- **Sign**: the signer id does the following
  - Choice of a pseudo-random integer $r \in \mathbb{Z}_l^*$, and computation of the public element $R' = rP'$.
  - Computation of the element of $\mathbb{G}(+) = \langle P \rangle$

$$\Sigma = S_1(id) + rH(id \| m) \tag{4.6}$$

  The couple (Σ, R') represents the signature by user id for message m.

- **Vrf**: the verifier does the following
  - Computation of digests $H(id)$ and $H(id \| m)$.
  - Check of the equality

$$e(\Sigma, P') = e(H(id), V')\, e(H(id \| m), R') \tag{4.7}$$

  and acceptance of the signature only if (4.7) is true.

Again, the correctness of the GS-IBS scheme easily follows from the bilinearity of e, since:

Proposition 4.2.1. *Suppose that no adversaries interfere with execution of signature scheme. For equation (4.7), the scheme is correct:*

$$\begin{aligned}
e(\Sigma, P') &= e(S_1(id) + rH(id\|m), P') \\
&= e(S_1(id), P')\, e(rH(id\|m), P') \\
&= e(H(id), V')\, e(H(id\|m), R')\ .
\end{aligned}$$

The following proposition, concerning the security of scheme GS-IBS, is a direct consequence of the results in [Gentry and Silverberg, 2002]:

Proposition 4.2.2. *Let $\mathbb{G} = \langle P \rangle$ and $\mathbb{G}' = \langle P' \rangle$, as returned by algorithm Setup, constitute a group pair of prime order l for which the co-CDH problem is hard to solve. Let the function $H : \{0,1\}^* \to \mathbb{G}$, returned by algorithm Setup, behave as a random oracle. Then scheme GS-IBS is secure against existential forgery under adaptive chosen-message attacks.*

4.3 Comparison of computational costs

The three IBS schemes previously considered are equivalent in terms of signature size, thus being roughly as bandwidth-efficient as the scheme [Barreto et al., 2005] (BLMQ-IBS), if point compression is put in place. Nevertheless, they differ in terms of computing costs. Table 4.1 shows operation counts for such schemes, distinguishing between signature and verification operations.

	PC	SM	PA	I	M	A	DC	RC
IBS-1 Sign	-	2	-	1	-	1	1	1
IBS-1 Vrf	2	1	1	-	-	-	2	-
IBS-2 Sign	-	2	-	1	-	1	1	1
IBS-2 Vrf	1	2	2	-	-	-	2	-
GS-IBS Sign	-	2	1	-	-	-	1	1
GS-IBS Vrf	3	-	-	-	1	-	2	-

Tab. 4.1: Comparison of computational costs among ID-based signature schemes IBS-1, IBS-2 and GS-IBS

Operations are denoted as acronyms of two-letter, as follows: PC = Paring Computations; SM = Scalar Point Multiplications; PA = Point Additions; I = Scalar Inversion ; M = Scalar Multiplications; A = Scalar Additions; DC = Digest Computations (i.e. evaluations of functions h or H); RC = Random number Computation. The comparison shows that the most efficient of the three schemes is IBS-2, which requires just one pairing computation. However, this gain in pairing computations among the considered schemes does not come for free as shown in Table 4.1, each pairing is offset by a point multiplication plus a point addition.

ID-based key agreement

In this Chapter, we present our contributions on ID-based key agreement protocols. According to the fact that WANETs communications can involve just two parties or a group of them, our work started from considering two notable protocols in the literature, one for two-party key agreement and the other for group key agreement. The first protocol was the natively ID-based one-round protocol of [Fiore and Gennaro, 2010], which can be implemented over any cyclic group of prime order where the CDH problem is supposed to be hard, without any bilinear structure. We propose here its elliptic curve version, named eFG, which requires just four scalar points multiplication and two point addition for each party. Instead, the group key agreement (GKA) protocol we started from was that of [Dutta and Barua, 2008] (DB), which in turn was derived by the famous unauthenticated protocol of [Burmester and Desmedt, 1994] (BD). The two GKA protocols discussed in the following Sections were both derived from elliptic curve version of the DB Protocol. Moreover, we introduce a set of procedures for the management dynamic groups of parties. Although these protocols can use a generic signature scheme, we coupled each of them with one of the ID-based signature schemes introduced in Chapter 4, in order to benefit of the advantages offered by the ID-based trust model in the context of WANETs, as described in Chapter 2. As we are going to show in the next Chapter, by coupling the GKA protocols introduced here with the ID-based signatures schemes of Chapter 4, we got authenticated GKA protocols that require small memory footprints and can achieve very low communication bandwidth and local computation costs for each of the involved parties.

5.1 eFG protocol

According to the ID-based enrollment procedure discussed in Section 3.4, protocol eFG requires the identifiers A and B of the two communicating parties plus all other parameters made public by the KGC. The first task of KGC is to define the elliptic curve $E(\mathbb{F}_q)$ to be used, and a generator P of a group $\mathbb{G}$ of prime order l, where p is the prime order of the field $\mathbb{F}_q$ and l is a big prime factor of $p-1$. Then, the KGC sets an hash function h with values in $\mathbb{Z}_l^*$ and a group to string map in $\mathbb{G} \times \mathbb{G}$. Finally, the KGC generates its private-public key pair s, V. The system public parameters are returned by algorithm eFG Setup, as follows:

eFG Setup

- the KGC returns the elliptic curve $E(\mathbb{F}_q)$ to be used.
- the point $P \in E$ of prime order l, which generates the cyclic group $\mathbb{G} \subset E$ where the Diffie-Hellmann system will be considered and the two hash functions.

$$h : \{0,1\}^* \to \mathbb{Z}_l^*, \quad H : \mathbb{G} \times \mathbb{G} \to \{0,1\}^n$$

  where n is a suitable positive integer.
- the public key $V = sP \in \mathbb{G}$ of the KGC, where $s \in \mathbb{Z}_l^*$ is the corresponding secret key, chosen randomly by the KGC.

Moreover, as result of an interaction with the KGC through a properly protected, out-of-band channel, each user receives its own private key related to its identity U. This key is the output of the KeyGen algorithm described in the following. It consists in a couple $(R_U, s_U) \in \mathbb{G} \times \mathbb{Z}_l^*$, which represents the elliptic version of the Schnorr's signature [Schnorr, 1991] of the string U under the public key V.

eFG KeyGen

KGC: $n_U \xleftarrow{\$} \mathbb{Z}_l^*$

$R_U = n_U P$

$s_U = n_U + sh(U \| R_U)$

$\xrightarrow{R_U, s_U} \{U\}$

Protocol eFG computations can be subdivided in two steps. In the first step, each user U asynchronously calculates its ephemeral private-public key couple $(t_U, T_U) \in \mathbb{Z}_l^* \times \mathbb{G}$. Then, each user U sends to the other party a packet with its identifier U alongside with the two points $R_U, T_U \in \mathbb{G}$. Finally, in the second step each user U computes a couple of points $(Z_U, Z'_U) \in \mathbb{G} \times \mathbb{G}$ and derives the key as $K = H(Z_U, Z'_U)$. The computation of (Z_U, Z'_U) requires the private information of U, together with V and the information received from its peer due to the previous communication step. Since each user computes the same couple (Z_U, Z'_U), the computed key K is actually the same for both users. It is worthwhile to note that the protocol actually does not rely on message authentication, since each of the two messages exchanged at first two are actually not signed by the sender. However, the way to derive (Z_U, Z'_U) assures that only the two parties with identities A and B can compute the key K.

eFG protocol

```
begin Round
    A: $t_A \xleftarrow{\$} \mathbb{Z}_l^*$; $T_A = t_A P$
    B: $t_B \xleftarrow{\$} \mathbb{Z}_l^*$; $T_B = t_B P$
    $A \xrightarrow{A,R_A,T_A} \{B\}$
    $A \xleftarrow{B,R_B,T_B} \{B\}$
end
begin KeyDerivation
    A: $Z_A = (t_A + s_A)(T_B + R_B + h(B||R_B)V)$; $Z_A^{'} = t_A T_B$
        $K = H(Z_A, Z_A^{'})$
    B: $Z_B = (t_B + s_B)(T_A + R_A + h(A||R_A)V)$; $Z_B^{'} = t_B T_A$
        $K = H(Z_B, Z_B^{'})$
end
```

It easy to show that protocol eFG is correct, meaning that it actually results in a key shared between users A and B.

Definition 5.1.1. *A key establishment protocol is correct if it results in a same, unique key shared among all the involved parties, assuming all of them execute the protocol in the right way and no adversary interferes with protocol execution.*

Proposition 5.1.1. *Suppose that parties A and B are honest and that no adversaries interfere with the execution of protocol eFG. Then, both A and B compute the same key K given by $K = H(Z_B, Z_B^{'})$*

Proof. The key expression $K = H(Z_U, Z_U^{'})$ is actually the same for $U = A$ and $U = B$ as shown by the following calculations:

$$
\begin{aligned}
Z_A &= (t_A + s_A)(T_B + R_B + h(B||R_B)V) \\
&= (t_A + n_A + sh(A||R_A))(t_B P + n_B P + h(B||R_B)sP) \\
&= (t_A + n_A + sh(A||R_A))((t_B + n_B + sh(B||R_B))P) \\
&= (t_A P + n_A P + sh(A||R_A)P)(t_B + s_B) \\
&= (T_A + R_A + h(A||R_A)V)(t_B + s_B) \\
&= Z_B
\end{aligned}
$$

$$
Z'_A = t_A T_B = t_A t_B P = T_A t_B P = Z'_B
$$

□

Protocol eFG is secure under the *strong Diffie-Hellmann assumption*, if the hash functions h and H behaves as random oracles (see theoretical notions in Chapter 3). This is a direct consequence of the fact that eFG is the elliptic version of protocol introduced in [Fiore and Gennaro, 2010], and that this last protocol was proven secure by their authors under the above assumptions. For each node, the eFG protocol requires 4 point-scalar multiplications, 2 point additions and 2 hash computations. Moreover, each node has to send and receive one message, which composes of the sender identifier and two points in $\mathbb{G}$.

	PC	SM	PA	I	M	A	DC	RC
eFG Protocol	-	4	2	-	-	-	2	-

Tab. 5.1: Operation required by each of the two parties involved in protocol eFG.

Operations in Table 5.1 are denoted as acronyms of two-letter, as follows: PC = Paring Computations; SM = Scalar Point Multiplications; PA = Point Additions;

I= Scalar Inversion; M = Scalar Multiplications; A = Scalar Additions; DC = Digest Computations (i.e. evaluations of functions h or H); RC = Random number Computation.

5.2 Extending the DH-protocol to n parties

A natural way to get an authenticated group key agreement protocol is through an extension of the DH exchange to more than two parties, coupled with a message signing procedure for assuring that protocol messages are actually from the intended parties and pertain to the current protocol session. As previously illustrated in Chapter 2, if correctly implemented, this approach has the advantage of inheriting from the DH protocol the security properties of key independence and perfect forward secrecy. Actually, the large majority of authenticated group key agreement (AGKA) protocols proposed up to now derive in some way from the DH protocol [Boyd and Mathuria, 2003],[Dutta and Barua, 2005],[Dutta and Barua, 2008],[Yao et al., 2008],[Joye and Neven, 2009].

Let us consider a cyclic (multiplicative) group of prime order l, and let g be one of its generators. Moreover, let $\mathbb{Z}_l^*$ denote the set of principal residues modulo l being co-prime to l, and let $n_i \in \mathbb{Z}_l^*$ be a secret chosen at random[1] by party M_i. A naive way to generalize the DH key agreement protocol to a set of m parties is to order them in a ring $\{M_i\}$, that is $M_i \neq M_j$ if $i \neq j \mod m$, and to perform $m-1$ communication rounds as follows. At round $j \quad (j = 1, \ldots, m-1)$, each M_i sends to its next party the DH exponential given by $\chi_{i,j} = \chi_{i-1,j-1}^{n_i}$, where $\chi_{i,0} = g$ for each i and all computations are performed modulo l.

This is precisely the [Ingemarsson et al., 1982] (ITW) protocol, which was perhaps the first attempt of group key agreement based on generalized DH exchanges. It should be clear that all parties after the $(m-1)$-th round can compute the same

[1] Actually, $\mathbb{Z}_l^*$ for a prime l consists of all the positive integers smaller than l, and $1 < n_i < l$ is usually calculated thanks to a *pseudo-random* bit generator.

session key, given by

$$k = g^{n_0 n_1 \dots n_{n-1}} \tag{5.1}$$

and that such protocol is secure against passive adversaries by assuming the hardness of the computational Diffie-Hellman (CDH) problem. The ITW protocol is very expensive: not only each party has to compute m modular exponentiations; it must also send and receive $m-1$ messages, for a total of $m(m-1)$ messages exchanged in $m-1$ rounds.

The three Group Diffie-Hellman protocols of [Steiner et al., 1996] are variations of the ITW protocol. They all compute the shared key given by equation (5.2), but differ in where computations are done and which messages are communicated. GDH.1 and GDH.2 protocol are both optimal with respect to the total number of messages sent by parties during a session, respectively for the case with and without broadcast (see [Becker and Wille, 1998] for an analysis of communication complexity bounds for GKA protocols). Indeed, GDH.1 attains the lowest bound of $2(m-1)$ messages, whilst GDH.2 requires m messages to be sent, one of which is a broadcast. However, all the GDH protocols require a number of rounds that linearly increase with m. Moreover, the total number of modular exponentiations is $\frac{m(m+3)}{2} - 1$ for GDH.1 and GDH.2, and $5m-6$ for GDH.3.

	ITW	GDH.1	GDH.2	GDH.3
Full-contributory	Yes	No	No	No
Rounds	$m-1$	$2(m-1)$	m	$m+1$
Total messages	$m(m-1)$	$2(m-1)$	m	$2m-1$
Total exponentiations	m^2	$\frac{m(m+3)}{2} - 1$	$\frac{m(m+3)}{2} - 1$	$5m-6$

Tab. 5.2: Comparing some group key-agreement protocols that generalize the DH two-party key exchange.

The AGKA protocols considered in [Bresson et al., 2001b],[Bresson et al., 2002] are all derived by GDH.2, thus becoming unpractical already for modest size groups of parties. However, the main contribution of these works is the first formal treatment of security for AGKA protocols. The model given in [Bresson et al., 2001b] was used to prove implicit key authentication and perfect forward secrecy in case of static groups of members, by assuming the intractability of the CDH problem and the the existence of a hash function behaving as a random oracle. This model has been successively extended to encompass dynamic groups [Bresson et al., 2001a] and strong corruption [Bresson et al., 2002], where a corruption other than long-term keys can reveal all not explicitly erased internal data. It has been adopted in various protocols by other authors in the following years, in order to prove the security of their AGKA proposals, and must be considered the reference model for the security of AGKA protocols.

5.3 Constant round protocols

A requisite for a scalable GKA protocol is to have a small number of rounds regardless the number of involved parties. Indeed, the running time of a protocol is proportional to the number of its communication rounds. Different GKA protocols have been proposed so far in the literature that attain a constant number of rounds, usually comprised between 2 and 3, but at the price of one or more of the following issues: expensive per node cryptographic operations which increase with the number of parties [Bresson and Catalano, 2004], high bit transfers over the network or big memory footprints for the involved parties, the burden and unbalance concerning the establishment of a group leader [Augot et al., 2007][Nam et al., 2004] or the lack of some important security property [Boyd and Nieto, 2002]. Moreover, many protocols assume knowledge of the authentic public-keys for all the parties involved into a session, as trust anchor point for the authenticated key agreement. This raises the problem of long-term public key management (i.e. key authentication, distribution and revocation) which although a once in issue could be extremely difficult or costly to solve for some environments and application scenarios.

A round-optimal AGKA protocol is [Boyd and Nieto, 2002], which requires just one round and is perhaps the most efficient authenticated group key agreement protocol ever designed. It makes use of hashed values instead of DH exponents, and was proven secure in the random oracle model. However that comes at the cost of some unbalance in computation efforts for parties, the initiator node being much more involved than the other nodes, and more important, the lack of the property of perfect forward secrecy.

Burmester-Desmedt (BD) proposed an unauthenticated group key agreement protocol [Burmester and Desmedt, 1994] which exchanges over the network also the second order DH exponentials, without exposing the factors $\chi_{i,2}$ of k. Each M_i, after receiving $g^{n_{i-1}}$ from M_{i-1} and $g^{n_{i+1}}$ from M_{i+1}, broadcasts to the other members the value $x_i = g^{(n_{i+1}-n_{i-1})n_i}$. Key k can be indeed computed by each M_i as:

$$k = g^{n_{i-1}n_i m} x_i^{m-1} x_{i+1}^{m-2} \dots x_{i+m-3}^{2} x_{i+m-2} \tag{5.2}$$

This broadcast version of the BD protocol requires two rounds, which is nearly optimal, since [Becker and Wille, 1998] gave the lowest bound of one single round for key agreement using broadcast. Computations involve a total of $m(m+1)$ modular exponentiations. The BD protocol was proven secure against passive attacks in the standard model by assuming the hardness of the CDH problem [Burmester and Desmedt, 1995]. Later, security against active adversaries for an authenticated version obtained through a general ”authentication compiler” was proven by [Katz and Yung, 2003], assuming the hardness of the DDH problem and the existence of a random oracle.

Dutta and Barua followed a more efficient approach [Dutta and Barua, 2008] in which K is computed as $K = k_0 k_1 \dots k_{m-1}$, and each M_i gets the $k_j (j = 0, \dots, m-1)$ due to the iterative formula:

$$\begin{aligned} k_i &= g^{n_{i-1}n_i} \\ k_{i+h} &= k_i \prod_{j=i}^{i+h-1} x_j \; (h = 1, \dots, m-1) \end{aligned} \tag{5.3}$$

in which indices are modulo m. With such a formula, computation of k requires one (full-length) modular exponentiation and $2m - 2$ modular multiplications ($m - 1$ multiplications to compute the k_i plus $m - 1$ multiplications to compute k from the k_i).

Formula (5.3) offers two other advantages with respect to computation of k through expression (5.2). First, each member M_i can detect if there was a misbehaving party by checking that $g^{n_{i+1}n_i} = k_{i+m-1}$. Second, precomputed data allows getting less overhead in the management of dynamic groups, for example groups where membership varies because of leaving or joining members. The authors got also an improvement in communication overhead for the authenticated version of the protocol. They consider as nonces to face replay attacks deterministic and unique instance numbers instead of the random values provided by the authentication compiler in [Katz and Yung, 2003], thus being able to reduce number of rounds by one. Their protocol was proven secure under the same hardness assumptions and adversarial model than [Katz and Yung, 2003].

5.4 GKA protocols

In this section we present two *GKA* protocols for authenticated group key agreement, that allow for implicit key authentication in two communication rounds. The proposed protocols turn out by combining the ID-based signature schemes described in Chapter 4 with variations of the DB protocol for elliptic curves. In practice, the way to compute the shared key is derived by iterative formula (5.3), rewritten in the elliptic curve arithmetic. Given an elliptic curve $E(\mathbb{F}_q)$, let $P \in E$ and $\mathbb{G} = \langle P \rangle$ be the additive group of prime order l generated by P on E. Moreover, for any integer i, let $[i]_m \in \{0, \dots, m-1\}$ denote the last non-negative residue of i modulo m. Then (5.3) can be rewritten as follows:

$$\begin{aligned} X_i &= n_{[i-1]_m} n_i P \qquad (i = 1, \dots, m-1; \quad n_{[i-1]_m}, n_i \in \mathbb{Z}_l^*) \\ X_{[i+h]_m} &= X_i + \sum_{j=i}^{i+h-1} (X_{[j+1]_m} - X_{[j]_m}) \quad (h = 1, \dots, m-1) \end{aligned} \tag{5.4}$$

and the shared key is given by

$$K = X_0 + X_1 + \dots + X_{m-1} \tag{5.5}$$

Actually, our protocols use a slight modification of (5.4), which allows to compute K using, just two communications among each party M_i and its adjacent nodes M_{i-1}, M_{i+1} at round one. In the sequel we refer to these protocols as GKA v1 and GKA v2, respectively.

Any multi party protocol requires a group initialization phase, in which each party of a suitable set of m members becomes aware of all other parties belonging to the multicast group and of an integer t. Such integer t uniquely identifies the communication session, and by incrementing in a suitable way its value, the different rounds during protocol execution.

After group initialization, GKA protocols order the set of participants in a lexicographic cyclic way $\{M_i\}$ $(i \in \{0, \dots, m-1\})$, and associate this ordered set to the identifier t. It is worth to stress here that a cyclic ordering among participants is required by each BD derived protocol, whilst the use of a round identifier is an approach often adopted in protocol design to counterfaith reply attacks and attacks caused by interleaving messages, both in the same session and among multiple concurrent sessions.

A natural way to achieve group initialization is due to a member acting as protocol initiator, that can be efficiently and securely achieved by splitting the first round of the protocols into a request-response interaction between the initiator and the recipients that agree in forming the group. The initialization and others procedures for GKA protocols are discussed in Section 5.5. Thus, we can assume for the moment that each party knows the ordered set $\{M_i\}$ $(i = 0, \dots, m-1)$ and the value of the identifier t. After that, authenticated key agreement is accomplished through the two message exchanges procedures Round1, Round2 and the local computation KeyDerivation.

5.4.1 Protocol GKA v1

Protocol GKA v1 is as follows:

- Round1: Each member M_i generates its ephemeral Diffie-Hellman couple (n_i, Z_i), where $Z_i = n_i P \in \mathbb{G} \backslash \{O\}$. M_i signs the binary string $m_i = \{j, Z_i\}$ with key s_i and the signing algorithm Sign. Finally, M_i sends (j, Z_i) along with the signature tag σ_i to nodes $M_{[i-1]_m}$ and $M_{[i+1]_m}$.

- Round2: Each member M_i verifies the authenticity of the two messages received from previous round thanks to the Vrf algorithm. If such verification succeeded, then M_i computes $X_i = n_i Z_{i+1}$, $Y_i = X_i - n_i Z_{i-1}$ and signs the string $m_i = \{t+1, Y_i\}$ to get the tag $\sum_i$. Afterwards, M_i broadcast $(t+1, Y_i)$ along with σ_i to the other group members. Otherwise, M_i broadcast the signed string corresponding to $t+1$ and stops, indicating that the signature verification failed.

- KeyDerivation: Each member M_i verifies the authenticity of the $m-2$ messages received from the previous round and that none of these message corresponds to the binary representation of $t+1$. If verification succeeded, then M_i computes X_j from Y_j $(j \neq i)$ and X_i through equation (5.4) and gets the session key $K = \sum_{j=0}^{m-1} X_j$. Otherwise, M_i performs a third communication round by multicasting the signed string corresponding to $(t+2)$ and stops, in order to indicate its failure in computing the session key.

Protocol GKA v1

Setup: A group $\mathbb{G} = \langle P \rangle$, where P is a point of prime order p over an elliptic curve.

Result: A secret multicast key K computed with the pairwise contribution of all members M_i.

```
begin Round1
    for $i \leftarrow 0$ to $m-1$ do
        $M_i :$ priv $n_i \overset{\$}{\leftarrow} \mathbb{Z}_l^*$, pub $Z_i = n_i P,\ t = \mathsf{Cnt}(\{M_i\}),\ \sigma_i = \mathsf{Sign}(\{t, Z_i\})$
        $M_i \xrightarrow{t, Z_i, \sigma_i} \{M_{[i-1]_m}, M_{[i+1]_m}\}$
    end
end
```

$\rightarrow$

```
$\leftarrow$

begin Round2
    for $i \leftarrow 0$ to $m-1$ do
        if $M_i :\ \mathsf{Vrf}(\sigma_{[i-1]_m}) \wedge \mathsf{Vrf}(\sigma_{[i+1]_m})$ then
            $M_i :\ X_i = n_i Z_{[i+1]_m}, Y_i = X_i - n_i Z_{[i-1]_m},\ \sigma_i =$
            $\mathsf{Sign}(\{t+1, Y_i\})$
            $M_i \xrightarrow{t+1, Y_i, \sigma_i} \{M_\iota :\ \iota \neq i\}$
        else
            $M_i :\ \sigma_i = \mathsf{Sign}(\{t+1\})$
            $M_i : \xrightarrow{t+1, \sigma_i} \{M_\iota :\ \iota \neq i\}\ M_i :$ exit
        end
    end
end
begin KeyDerivation
    for $i \leftarrow 0$ to $m-1$ do
        if $M_i :\ \bigwedge_{\iota \neq i} (\mathsf{Vrf}(m_\iota, \sigma_\iota, M_\iota) \wedge \mathtt{msg}_\iota \neq \{t+1\})$ then
            for $h \leftarrow 1$ to $m-1$ do
                $M_i :\ X_{[i+h]_m} = X_{[i+h-1]_m} + Y_{[i+h]_m}$
            end
            if $M_i :\ X_{[i+m-1]_m} = n_i Z_{[i-1]_m}$ then
                $M_i :$ priv $K_i = X_0 + X_1 + \ldots + X_{m-1}$
            else
                $M_i :\ \sigma_i = \mathsf{Sign}(\{t+2\})$
                $M_i \xrightarrow{t+2, \sigma_i} \{M_\iota :\ \iota \neq i\}\ M_i :$ exit
            end
        end
    end
end
```

5.4.2 Protocol GKA v2

In protocol GKA v1, the number of signature verifications performed by each party increases linearly with the number m of parties. That can result in significant computational and power costs in case of suitably large groups of parties, preventing the use of this protocol in some applications scenarios. Since bilinear pairing computations are much more expensive then others elliptic operations, the above is particularly true for ID-based signature schemes as those considered in Chapter 4. Indeed, such schemes get shorter signatures at the price of bilinear pairing computations during the verification process. Protocol GKA v2 uses a cooperative checking of the values transferred among parties in order to avoid the signing of messages at Round two and their verification during the key derivation phase. This allows to avoid for each party a signature generation and $m-1$ signature verifications, saving local operations costs and network bandwidth.

Protocol GKA v2

Setup: A group $\mathbb{G} = \langle P \rangle$, where P is a point of prime order p over an elliptic curve.

Result: A secret multicast key K computed with the pairwise contribution of all members M_i.

```
begin Round1
    for $i \leftarrow 0$ to $m-1$ do
        $M_i : \ n_i \overset{\$}{\leftarrow} \mathbb{Z}_p^*, \ Z_i = n_i P, \ \sigma_i = \mathsf{Sign}(\{t, Z_i\})$
        $M_i \ \xrightarrow{t, Z_i, \sigma_i} \ \{M_{[i-1]_m}, M_{[i+1]_m}\}$
    end
end
```

$\rightarrow$

```
$\leftarrow$

begin Round2
    for $i \leftarrow 0$ to $m-1$ do
        if $M_i :\ \mathsf{Vrf}(\sigma_{[i-1]_m}) \wedge \mathsf{Vrf}(\sigma_{[i+1]_m})$ then
            $M_i :\ X_i = n_i Z_{[i+1]_m}, Y_i = X_i - n_i Z_{[i-1]_m}$
            $M_i \xrightarrow{Y_i} \{M_\iota :\ \iota \neq i\}$
        else
            $M_i :\ \sigma_i = \mathsf{Sign}(\{t+1\})$
            $M_i \xrightarrow{t+1,\sigma_i} \{M_\iota :\ \iota \neq i\}$ $M_i :$ exit
        end
    end
end
begin KeyDerivation
    for $i \leftarrow 0$ to $m-1$ do
        for $h \leftarrow 1$ to $m-1$ do
            $M_i :\ X_{[i+h]_m} = X_{[i+h-1]_m} + Y_{[i+h]_m}$
        end
        if $M_i :\ X_{[i+m-1]_m} = n_i Z_{[i-1]_m}$ then
            $M_i :$ priv $K_i = X_0 + X_1 + \ldots + X_{m-1}$
        else
            $M_i :\ \sigma_i = \mathsf{Sign}(\{t+2\})$
            $M_i \xrightarrow{t+2,\sigma_i} \{M_\iota :\ \iota \neq i\}$ $M_i :$ exit
        end
    end
end
```

- Round1 performs the same operation of GKA v1.
- Round2: M_i verifies the authenticity of the two messages received from the previous round thanks to the Vrf algorithm. If such verification succeeded, then M_i computes $X_i = n_i Z_{[i+1]_m}$, $Y_i = X_i - n_i Z_{[i-1]_m}$. Afterwards, M_i multicasts Y_i to the other group members. Otherwise, M_i multicasts the signed string corresponding to the binary representation of $t+1$ and stops, indicating its failure in authenticating one of its neighbours.
- KeyDerivation: M_i, using the points Y_j $(j \neq i)$ received from the previous round, computes iteratively $X_{[i+h]_m}$ $(h = 1, \ldots, m-1)$ from X_i due to (5.4). Afterwards, it checks if point $X_{[i+m-1]_m}$ obtained is equal to point $n_i Z_{[i-1]_m}$ computed at round Round2. If that is the case, M_i gets the session key $K = \sum_{j=0}^{m-1} X_j$. Otherwise, M_i performs a third communication round by multicasting the signed string corresponding to $(t+2)$ and stops, in order to indicate its failure in computing the session key.

5.4.3 Comparison of computational costs

Table 5.3 shows a performance comparison among the different authenticated GKA protocols obtained by coupling GKA v1 and GKA v2 with one of the signature schemes IBS-1, IBS-2 and GS-IBS. Computational costs are relative to a single party, and take into account only pairing computations and scalar multiplications. Other operations (i.e. point addition, hashing, random number generation, sending and receiving messages) were omitted, because they result on similar overheads for both protocols. For example, both GKA v1 and GKA v2 send and receive (m + 1) messages.

Protocol / Scheme	Pairing Computation	Scalar Multiplication
GKA v1 / IBS-1	$2(m+1)$	$m+8$
GKA v2 / IBS-1	4	7
GKA v1 / IBS-2	$(m+1)$	$2m+9$
GKA v2 / IBS-2	2	9
GKA v1 / GS-IBS	$3(m+1)$	7
GKA v2 / GS-IBS	6	5

Tab. 5.3: Comparison of computational costs among the authenticated group key agreement protocols discussed in this Chapter.

5.4.4 Run example

As an illustrative example of GKA v1 protocol run, let us consider the case of four members $\{M_i\}$, $(i = 0, 1, 2, 3)$.

At Round 1, each member M_i choices randomly a secret exponent $n_i \in \mathbb{Z}_l^*$, computes point $Z_i = n_i P$, and gets through algorithm Sign signature σ_i of the string corresponding to the concatenation of the binary representations of t and $Z(i)$. Then the following communications occur:

$$\begin{aligned}
M_0 : &\quad \xrightarrow{t,Z_0,\sigma_0} \{M_3, M_1\} \\
M_1 : &\quad \xrightarrow{t,Z_1,\sigma_1} \{M_0, M_2\} \\
M_2 : &\quad \xrightarrow{t,Z_2,\sigma_2} \{M_1, M_3\} \\
M_3 : &\quad \xrightarrow{t,Z_3,\sigma_3} \{M_2, M_0\}
\end{aligned}$$

At Round 2, each member M_i verifies the authenticity of the two messages it received from Round 1, computing $X_i = n_i Z_{i+1}$ and $Y_i = X_i - n_i Z_{i-1} = X_i - X_{i-1}$ if such verification succeeds. Then M_i the signature σ_i of the string corresponding to the concatenation of the binary representations of $t+1$ and Y_i (signature at this Round were named in the same way, since they supersede those of Round 1). Then the following communications occur:

$$\begin{aligned}
M_0 : &\quad \xrightarrow{t+1, Y_0, \sigma_0} \{M_1, M_2, M_3\} \\
M_1 : &\quad \xrightarrow{t+1, Y_1, \sigma_1} \{M_2, M_3, M_0\} \\
M_2 : &\quad \xrightarrow{t+1, Y_2, \sigma_2} \{M_3, M_0, M_1\} \\
M_3 : &\quad \xrightarrow{t+1, Y_3, \sigma_3} \{M_0, M_1, M_2\}
\end{aligned}$$

After a successful verification through algorithm Vrf of the authenticity of messages received from the other three members, each M_i computes points X_j $(j \neq i)$ as follows:

$$\begin{aligned}
M_0 : &\quad X_1 = X_0 + Y_1,\ X_2 = X_1 + Y_2,\ X_3 = X_2 + Y_3 \\
M_1 : &\quad X_2 = X_1 + Y_2,\ X_3 = X_2 + Y_3,\ X_0 = X_3 + Y_0 \\
M_2 : &\quad X_3 = X_2 + Y_3,\ X_0 = X_3 + Y_0,\ X_1 = X_0 + Y_1 \\
M_3 : &\quad X_0 = X_3 + Y_0,\ X_1 = X_0 + Y_1,\ X_2 = X_1 + Y_2
\end{aligned}$$

After which, each node can compute the session key $K = X_0 + X_1 + X_2 + X_3$.

Things go in the same way for protocol GKA v2, except that in this case the broadcast messages sent at Round 2 are not signed by the sender and are not verify by their receivers. Rather, after computing the $X_j (j \neq i)$ as in protocol GKA v1, each member does the following respective checks:

$$
\begin{aligned}
M_0 :& \quad X_3 \overset{?}{=} n_0 Z_3 \\
M_1 :& \quad X_0 \overset{?}{=} n_1 Z_0 \\
M_2 :& \quad X_1 \overset{?}{=} n_2 Z_1 \\
M_3 :& \quad X_2 \overset{?}{=} n_3 Z_2
\end{aligned}
$$

If the check is passed and the member does not receive messages from any other member (meaning that all checks were passed), then it assumes that the computed key $K = X_0 + X_1 + X_2 + X_3$ is authentic.

5.4.5 Correctness and security

In this Section, we discuss the correctness and the security of GKA protocols. The correctness property assures that normal executions of a given protocol always terminate and return the expected outcomes. In the case of protocol GKA v1 and GKA v2 it suffices to prove that each involved party computes the same key K given by (5.4). That is stated in the following

Proposition 5.4.1. *Suppose that all parties $\{M_i\}$ $(i \in \{0, \ldots, m-1\})$ involved in protocol GKA v1 (GKA v2) are honest and that no adversaries interfere with protocol execution. Then, each M_i computes a unique K_i such that $K_i = K_j$ for any* $0 \leq i < j \leq m-1$

Proof. Under the given assumptions, at round one of both protocols GKA v1 and GKA v2 each M_i computes $Z_i = n_i P$, and receives $Y_{[j]_m} = X_{[j]_m} - n_{[j]_m} Z_{[j-1]_m}$ $j \in \{i-1, i+1\}$.
Thus, starting from $j = i+1$, each M_i can iteratively compute $X_{[j]_m} = n_{[j]_m} Z_{[j-1]_m} + Y_{[j]_m} = n_{[j]_m} n_{[j+1]_m} P$ for each $j \neq i$. Moreover, since M_i knows the secret n_i and at round one has received $Z_{[i+1]_m} = n_{[i+1]_m} P$, it can compute $X_i = n_i Z_{[i+1]_m} = n_i n_{[i+1]_m} P$. Thus M_i terminates by computing

$$K_i = \sum_{j=0}^{m-1} X_j = (n_0 n_1 + n_1 n_2 + \cdots + n_{m-2} n_{m-1} + n_{m-1} n_0)P \tag{5.6}$$

which is the same point regardless of the value of the index i. □

The sequel of this Section deals with the security of GKA protocols. As we told in Chapter 2, the main notion of security for key establishment protocols is that of (implicit) key authentication. However, it is important to stress here that this notion actually comes in two flavors in the literature, that we denote here as *computationally-hard* and *semantically-hard* key authentication, respectively. The first notion establishes that an adversary cannot *compute* the key, whilst the second notion requires that no information at all can be derived by the adversary about the key.

Definition 5.4.1. *A GKA protocol is said to have the property of implicit key authentication if each player is assured that no other players from the arbitrary pool of players can compute the session key.*

Definition 5.4.2. *A GKA protocol is said to have the property of implicit key authentication if each player is assured that no other players from the arbitrary pool of players can learn any information about the session key.*

If each player is moreover assured that its parties actually have possession of the same key, the protocol is said to have the property of *explicit key authentication*. The second Definition, introduced in [Bellare et al., 1998] for two party protocols and in [Bresson et al., 2001b] for multi-party protocols, is a semantically oriented notion of security [Goldwasser and Micali, 1982]. It is clearly more strong than the first one, in the sense that if a protocol satisfies Definition 5.4.1 then it must satisfy Definition 5.4.2, too.

The following propositions are about the security of the GKA protocols introduced in Section 5.4. It is worth to note at this point that protocols very similar to GKA v1 were already proved to have the property of implicit key authentication [Katz and Yung, 2007], [Dutta et al., 2004]. However, the above proofs are relative to the semantically-hard notion of implicit key authentication, and require the *DDH hardness assumption* plus the existence of a random oracle.
On the other hand, the signature schemes introduced in Chapter 4 are based on a gap between the difficulty of the DDH problem and the CDH problem, i.e. they work on groups where the DDH is easy to solve but the CDH is not. Thus, our proofs must work under the weaker assumption of the hardness of the CDH, and can moreover be conducted in the standard model. The payoff for these less stringent assumptions is that our proofs allow to deduce the weaker goal of computationally-hard key authentication given by Definition 5.4.1. However, keys are not semantically meaningful text messages but rather strings of randomly chosen bits. Thus, by assuming that keys are obtained through a cryptographically secure pseudo-random generator, the two notions of key authentication should actually be equivalent. This is confirmed by the fact that an heuristic to achieve property (5.4.2) from a key K satisfying (5.4.1) is to consider the key given by $H(K)$, where H is a pseudo-random function [Burmester and Desmedt, 2005].

Proposition 5.4.2. *The authenticated protocols obtained by coupling GKA v1 with one of the signature schemes introduced in Chapter 4 achieve perfect forward secrecy and computationally-hard implicit key authentication under CDH hardness assumption. Moreover, under the extra hypothesis that radio jamming* [2] *is detected by each legitimate party and results in aborting the protocol session, such protocols achieve explicit key authentication.*

Proof. Let $\{M_i\}$ $(i \in \{0, \dots, m-1\})$ denote the set of intended participants. Since

[2] Radio jamming works by the transmission of radio signals that disrupt communications by decreasing the signal-to-noise ratio. It is usually easy to detect because it can be heard on the receiving equipment [Berg, 2008]

the M_i behave honestly, an attacker $A \neq M_i$ can get the key K given by (5.5) using only one of the following ways:

1. Computing K from the messages exchanged by the M_i over the network.

2. Impersonating one or more of the M_i, so that the protocol actually runs among a subset of M_i and one or more A's instances.

3. Altering protocol execution in a way that he/she can derive the current session key.

Case (2) is ruled out by the fact that each message sent over the network is signed by the legitimate party, and that the signature schemes of 4 have the property of existential unforgeability [Goldwasser et al., 1988]. Indeed, in order to impersonate one or more of the M_i, A should be able to send at least one authenticated message pretending to be one of the M_i. Case (3) is ruled out by similar reasoning since no messages can be altered in any way, and each messages is uniquely binded to the current protocol session through its signature and the session identifier t. Ad for case (1), suppose A got K from eavesdropping the values Z_i and Y_i. From (5.4) and (5.5) it follows that

$$K = X_j + \sum_{h \neq j} X_h = mX_j + S(Y)$$

where S(Y) is a quantity depending from Y_i $(i \neq j)$ only. Thus A knows $mX_j = K - S(Y)$ for at least one $j \in \{0, \ldots, m-1\}$. However, this would require A to solve the CDH problem $X_j = n_{j-i}Z_j$, which by the contrary is assumed to be hard. Finally, the perfect forward secrecy is a consequence of the fact that K is obtained from a DH type exchange. Actually, protocols implementing such exchange achieve the stronger property of *key independence* [Boyd and Mathuria, 2003].

Under the extra hypothesis of radio jamming detection, each party has corroborate evidence that, if it has not received any exception message, then all other parties actually computed the same key, as prescribed by Proposition 5.4.1. Indeed, all messages exchanged over the network can neither be altered nor blocked □

Proposition 5.4.3. *The authenticated protocols obtained by coupling GKA v2 with one of the signature schemes introduced in Chapter 4 achieve perfect forward secrecy and computationally-hard explicit key authentication under the CDH hardness assumption and and the extra hypothesis that radio jamming is detected by each of the involved party and results in aborting the protocol session.*

Proof. We will show that under the extra hypothesis of jamming detection the unsigned messages Y_i broadcasted at round two must coincide with the ones computed by their legitimate senders. If that is the case, then all messages exchanged through GKA v2 are actually unforgeable as in protocol GKA v1, and the statement follows form Proposition 5.4.2. Thus, let us suppose that the adversary A changes one or more values Y_i broadcasted by legitimate nodes at round two, and let $\widetilde{Y}_i$ denote these possibly forged values. Since the exception messages broadcasted by each party at round two are signed, jamming is the only way A has to avoid that any such message reaches the other involved parties. However, jamming is detected. Thus, it suffices to prove that if neither exception messages are received by node M_i nor it detects a jamming attack, and if moreover the check provided at round two of protocol GKA v2.

$$X_{[i+m-1]_m} \overset{?}{=} n_i Z_{[i-1]_m} \tag{5.7}$$

is passed, then M_i has corroborate evidence that $\tilde{Y}_j = Y_j$ for each $j \in \{0, \ldots, m-1\}$. If check (5.7) is passed, then M_i has evidence that

$$\sum_{h=1}^{m-1} \tilde{Y}_{[i+h]_m} = \sum_{h=1}^{m-1} Y_{[i+h]_m} \tag{5.8}$$

since it received the authentic $Z_{[i-1]_m}$ and $Z_{[i+1]_m}$ at round one. On the other hand, since parties are honest and neither exception messages were received by M_i nor it detected jamming attacks, then M_i must likewise assume that

$$\sum_{h=1}^{m-1} \tilde{Y}_{[j+h]_m} = \sum_{h=1}^{m-1} Y_{[j+h]_m} \quad (j \neq i) \tag{5.9}$$

Thus we obtain the set of m relations

$$\sum_{h=1}^{m-1} \tilde{Y}_{[j+h]_m} = \sum_{h=1}^{m-1} Y_{[j+h]_m} \quad (j = i, \ldots, m-1, 0, \ldots, i-1)$$

and, by subtracting each one with its next, we get

$$\begin{array}{rcl} \tilde{Y}_{[i+1]_m} - \tilde{Y}_i & = & Y_{[i+1]_m} - Y_i \\ \tilde{Y}_{[i+2]_m} - \tilde{Y}_{[i+1]_m} & = & Y_{[i+2]_m} - Y_{[i+1]_m} \\ \ldots\ldots & & \ldots\ldots \\ \tilde{Y}_i - \tilde{Y}_{[i-1]_m} & = & Y_i - Y_{[i-1]_m} \end{array}$$

Since for node M_i by construction $\tilde{Y}_i = Y_i$, then the claim follows by applying recursively the above equalities.
One more consequence of the hypothesis concerning radio jamming detection is that each party has corroborate evidence that, if it has not received any exception message, then all other parties actually computed the same key, as prescribed by Proposition 5.4.1. Thus, protocol GKA v2 achieves explicit key authentication. □

5.5 GKA dynamic procedures

We describe a set of procedures for GKA protocol, designed to manage dynamic group of members. These procedures allow respectively for *group creation*, *key* *updating*, *joining* and *leaving*.

5.5.1 Group creation and key updating

Group creation is a basic operation for group communications, whose main task is to associate to each group a unique (valid) shared session key. As we said, GKA protocol group creation requires the arrangement of group members into a ring, which results in defining the effective set $\mathcal{M}$ of participants, establishing a lexicographic cyclic order for $\mathcal{M} = \{M_i\}$ $(i \in \{0, \ldots, m-1\})$, and associating to this ordered set an unique element t as group identifier. In such respect, it is worthwhile to assume that a group of members comes to life when its shared key is created, and ceases to be when such key is expired or revoked. Thus, it is proper to associate a group and its key to the same identifier, and to allow just one identifier for each group. However, in case of key updating, it seems proper also taking the same group identifier but changing the key identifier, in order to account for the key change. As shown in what follows, we solve these two seemingly conflicting requirements by setting both identifiers to the same integer at group creation, then incrementing it as protocol rounds and sessions go on. This way, the same integer serves both as group and round/session identifier. A natural way to obtain group initialization is through a request-response interaction between a member M, acting as *protocol initiator*, and a set $\mathcal{M}'$ of recipients representing a (possibly proper) superset of $\mathcal{M} \backslash M$. That can be efficiently achieved by reformulating $\mathsf{Round1}$ of protocol (5.4.1) in the following two rounds:

Create request M computes an ephemeral DH couple $(n_M, Z_M) \in (\mathbb{Z}_l^*, \mathbb{G})$ and the integer $t = \mathsf{Cnt}(Z_M)$, where Cnt is a public known function. Then it signs the string $\{t, Z_M\}$ and multicast t and Z_M along with the signature to the set of members $\mathcal{M}' \supseteq \mathcal{M} \backslash M$ with which it intends to form a group.

Create response If a recipient $S \in \mathcal{M}'$ does not agree in belonging to the group, then it discards the received packet (and all packets received form the other members in $\mathcal{M}'$ as replies to M query). Otherwise, after a successful verification of M signature, S computes $t = \mathsf{Cnt}(Z_M)$ and its ephemeral DH couple $(n_S, Z_S) \in (\mathbb{Z}_l^*, \mathbb{G})$, signs the string $\{t, Z_S\}$ with its own signature key, and then multicasts t and Z_S along with the signature to $\mathcal{M}'$.

It should be clear that, after the above two rounds, each member in $\mathcal{M}$ can establish which are the other members belonging to $\mathcal{M}$, and compute the group identifier through the public function Cnt. Thus, by ordering lexicographically the identities in $\mathcal{M}$, each member can order $\mathcal{M} = \{M_i\}$ $(i \in \{0, \dots, m-1\})$. This way, each party in $\mathcal{M}$ can run the two last steps of protocol GKA and compute the group key K, resulting in the following *Create* procedure.

Create Procedure

Summary: A member M (the protocol initiator) asks a set $\mathcal{M}'$ of members. This way, a cyclic lexicographically ordered group $\mathcal{M} \subseteq \mathcal{M}'$ which include M is established, and a shared secret key is univocally associated to the group.

Result: The set $\mathcal{M} = \{M_i\}$ $(i = 0, \dots, n-1;\ n > 1)$ and its unique identifier t, with $M \in \mathcal{M}$. A secret shared key K computed with the pairwise contribution of all M_i.

```
begin Create request
    $M:\ n_M \xleftarrow{\$} \mathbb{Z}_l^*,\ Z_M = n_M P,\ t = \mathsf{Cnt}(\{Z_M\}),\ \sigma_M = \mathsf{Sign}(\{t, Z_M\})$
    $M \xrightarrow{t, Z_M, \sigma_M} \mathcal{M}' \backslash M$
end
```

```
begin Create response
    for $S \in \mathcal{M} \backslash M$ do
        if $S:$ $\mathsf{Vrf}(\sigma_M)$ then
            $S:\ n_S \overset{\$}{\leftarrow} \mathbb{Z}_l^*,\ Z_S = n_S P,\ j = \mathsf{Cnt}(\{Z_M\}),\ \sigma_S =$
            $\mathsf{Sign}(\{t, Z_S\})$
            $S \xrightarrow{t, Z_S, \sigma_S} \mathcal{M}'$
        else
            $S:$ exit
        end
    end
end
begin Group key computation
    if $\mathsf{Round2}(j, \mathcal{M})$ then
        $\mathsf{KeyDerivation}(t+1, \mathcal{M})$
    end
end
```

Key updating occurs when a previous group key is revoked and the same group needs another key to continue its operations. Updating a group key provides better security in case of long sessions, and it is necessary in case a current key is compromised. In the following we consider only the simplest case where key revocation is asked by a member of the same group, and not discuss how such revocation takes place. Although very important in real world scenarios, a comprehensive analysis of key revocation requires indeed tools that go beyond cryptography and may strictly depend on the considered application scenario. Thus, we assume here that if a node asks for a new group key that request is mandatory to the other group members.
Key updating procedure, coherently with the assumption that legitimate nodes behaves honestly, implements such kind of interaction. It consists of an upgrade request

procedure in which the requester $M_k \in \mathcal{M}$ computes new cryptographic material, signs its public components and multicasts them alongside its signature to the other group members. Then an update response procedure takes place, in which only the two neighbours $M_{[k-1]_m}, M_{[k+1]_m} \in \mathcal{M}$ have to compute and send updated public cryptographic material. The protocol ends with a call to procedure KeyDerivation with new inputs from nodes $M_{[k-1]_m}, M_k$ and $M_{[k+1]_m}$.

Key Update Procedure

Summary: A member $M_k \in \mathcal{M}$ (the protocol initiator) asks the other members to update the key associated to $\mathcal{M}$. This way, the old key is revoked, and a new shared secret key is computed from the old one and univocally associated to the group through its identifier.

Setup: The current key K and identifier t for group $\mathcal{M}$.

Result: A new secret multicast key K', computed from the old one K with the pairwise contribution of all $M \in \mathcal{M}$ and binded to the value of t.

```
begin Update request
    $M_k :\ n'_k \xleftarrow{\$} \mathbb{Z}_l^*,\ Z'_k = n'_k P,\ X'_k = n'_k Z_{[k+1]_m},\ Y'_k =$
    $X'_k - n'_k Z_{[k-1]_m},\ \sigma_k = \mathsf{Sign}(\{t+1, Z'_k, Y'_k\})$
    $M_k \xrightarrow{t+1, Z'_k, Y'_k, \sigma_k} \mathcal{M} \backslash M_k$
end
```

```
begin Update response
    for $S \in \mathcal{M} \backslash M_k$ do
        if $S:\ \mathsf{Vrf}(\sigma_k)$ then
            if $S = M_{[k-1]_m}$ then
                $S:\ X'_{[k-1]_m} = n_{[k-1]_m} Z'_k, Y'_{[k-1]_m} =$
                $X'_{[k-1]_m} - n_{[k-1]_m} Z_{[k-2]_m}$
                $S \xrightarrow{Y'_{[k-1]_m}} \{M_\iota :\ \iota \neq [k-1]_m\}$
            end
            if $S = M_{[k+1]_m}$ then
                $S:\ Y'_{[k+1]_m} = X_{[k+1]_m} - n_{[k+1]_m} Z'_k$
                $S \xrightarrow{Y'_{[k+1]_m}} \{M_\iota :\ \iota \neq [k+1]_m\}$
            end
        else
            $S:$ exit
        end
    end
end
begin Key update
    $\mathsf{KeyDerivation}(Y_0, \ldots, Y'_{[k-1]_m}, Y'_k, Y'_{[k+1]_m}, \ldots, Y_{m-1})$
end
```

5.5.2 Group joining and leaving

Group modifications occur when a single member join or leave a previously established multicast group. We assume that the joining and leaving of single members do not affect the identity of a group. In other words, addition or subtraction of single participants leave the group identifier unaffected. This is because group modification, likewise key updating, can be performed through lightweight procedures which preserve most of the cryptographic material employed during group creation. The *Join* procedure implements the case of a member M joining a group $\mathcal{M}$ of which it knows all members. This way, M can multicast its join request to all the intended recipients, and can determine the ordered set $\{M_i\} = \mathcal{M} \cup \{M\}$ and the index k such that $M = M_k$. If that were not the case, M could easily get such information by interacting with anyone of the group members, without any cryptographic operation. Procedure resembles the key updating procedure, but there are two main differences:

- M cannot send in the join request the 2nd order public element Y_M, since it does not know the 1st order public elements of its neighbours.
- nodes in $\mathcal{M}$ have to compute the new cycling ordering of the group to encompass the entry of M.

Join Procedure

Summary:

A member M (the protocol initiator) asks to join the multicast group $\mathcal{M}$. A new shared secret key is univocally associated to the ordered group $\mathcal{M}' = \mathcal{M} \cup \{M\} = \{M_i,\}$ where $M = M_k$, in such a way that computations related to procedures Round1 and Round2 for members of $\mathcal{M}$ are preserved.

Result: A new secret multicast key K' for the group $\mathcal{M}'$, computed from the old key K of $\mathcal{M}$ with the pairwise contribution of all members in $\mathcal{M}'$.

```
begin Join request
    $M_k : \; n_k \overset{\$}{\leftarrow} \mathbb{Z}_l^*, \; Z_k = n_k P, \; \sigma_k = \mathsf{Sign}(\{Z_k\}) \; M_k \xrightarrow{Z_k, \sigma_k} \mathcal{M}$
end
```

```
begin Join response
    for $S \in \mathcal{M} \backslash M_k$ do
        if $S:$ $\mathsf{Vrf}(\sigma_k)$ then
            if $S = M_{[k-1]_m}$ then
                $S:$ $\sigma_{[k-1]_m} = \mathsf{Sign}(\{t+1, Z_{[k-1]_m}\})$
                $S \xrightarrow{t+1, Z_{[k-1]_m}, \sigma_{[k-1]_m}} M_k$
            end
            if $S = M_{[k+1]_m}$ then
                $S:$ $\sigma_{[k+1]_m} = \mathsf{Sign}(\{t+1, Z_{[k+1]_m}\})$
                $S \xrightarrow{t+1, Z_{[k+1]_m}, \sigma_{[k+1]_m}} M_k$
            end
        else
            $S:$ exit
        end
    end
end
begin Round2 update
    if $S = M_{[k-1]_m}$ then
        if $S:$ $\mathsf{Vrf}(\sigma_k)$ then
            $S:$ $X'_{[k-1]_m} = n_{[k-1]_m} Z_k, Y'_{[k-1]_m} =$
            $X'_{[k-1]_m} - n_{[k-1]_m} Z_{[k-2]_m}$ $S \xrightarrow{Y'_{[k-1]_m}} \{M_\iota : \iota \neq [k-1]_m\}$
        else
            $S:$ $\sigma_{[k-1]_m} = \mathsf{Sign}(\{t+2\})$
            $S \xrightarrow{t+2, \sigma_{[k-1]_m}} \{M_\iota : \iota \neq [k-1]_m\}$
        end
    end
end
```

```
if $S = M_k$ then
    if $S:\ \mathsf{Vrf}(\sigma_{[k-1]_m}) \wedge \mathsf{Vrf}(\sigma_{[k+1]_m})$ then
        $S:\ X_k = n_k Z_{[k+1]_m}, Y_k = X_k - n_k Z_{[k-1]_m}\ S \xrightarrow{Y_k} \{M_\iota :\ \iota \neq k\}$
    else
        $S:\ \sigma_k = \mathsf{Sign}(\{t+2\})\ S \xrightarrow{t+2,\sigma_k} \{M_\iota :\ \iota \neq k\}$
    end
end
if $S = M_{[k+1]_m}$ then
    if $S:\ \mathsf{Vrf}(\sigma_k)$ then
        $S:\ Y'_{[k+1]_m} = X_{[k+1]_m} - n_{[k+1]_m} Z_k$
        $S \xrightarrow{Y'_{[k+1]_m}} \{M_\iota :\ \iota \neq [k+1]_m\}$
    else
        $S:\ \sigma_{[k+1]_m} = \mathsf{Sign}(\{j+2\})$
        $S \xrightarrow{t+2,\sigma_{[k+1]_m}} \{M_\iota :\ \iota \neq [k+1]_m\}$
    end
end
begin Key update
    $\mathsf{KeyDerivation}(Y_0, \ldots, Y'_{[k-1]_m}, Y_k, Y'_{[k+1]_m}, \ldots, Y_{m-1})$
end
```

Leave Procedure

Summary:

A member $M_k \in \mathcal{M}$ (the protocol initiator) asks to leave the multicast group $\mathcal{M}$. A new shared secret key is univocally associated to the group $\mathcal{M}' = \mathcal{M} \backslash M_k$ in such a way that computations related to procedures Round1 and Round2 for members of $\mathcal{M}'$ are preserved.

Result: A secret multicast key K' for $\mathcal{M}'$, computed from the old one of $\mathcal{M}$ with the pairwise contribution of all members in $\mathcal{M}'$.

```
begin Leave request
    $M_k :\ \sigma_k = \mathsf{Sign}(\{j+1\})$
    $M_k \xrightarrow{\sigma_k} \mathcal{M}\backslash M_k$
end
begin Leave response
    for $S \in \mathcal{M}\backslash M_k$ do
        if $S :\ \mathsf{Vrf}(\sigma_k)$ then
            if $S = M_{[k-1]_m}$ then
                $S :\ \sigma_{[k-1]_m} = \mathsf{Sign}(\{t+1, Z_{[k-1]_m}\})$
                $S \xrightarrow{t+1, Z_{[k-1]_m}, \sigma_{[k-1]_m}} M_{[k+1]_m}$
            end
            if $S = M_{[k+1]_m}$ then
                $S :\ \sigma_{[k+1]_m} = \mathsf{Sign}(\{t+1, Z_{[k+1]_m}\})$
                $S \xrightarrow{t+1, Z_{[k+1]_m}, \sigma_{[k+1]_m}} M_{[k-1]_m}$
            end
        else
            $S :$ exit
        end
    end
end
```

```
begin Round2 update
    if $S = M_{[k-1]_m}$ then
        if $S:\ \mathsf{Vrf}(\sigma_{[k+1]_m})$ then
            $S:\ X'_{[k-1]_m} = n_{[k-1]_m} Z_{[k+1]_m}, Y'_{[k-1]_m} =$
            $X_{[k-1]_m} - n_{[k-1]_m} Z_{[k-2]_m}$
            $S \xrightarrow{Y'_{[k-1]_m}} \{M_\iota :\ \iota \neq [k-1]_m\}$
        else
            $S:\ \sigma_{[k-1]_m} = \mathsf{Sign}(\{t+2\})$
            $S \xrightarrow{t+2, \sigma_{[k-1]_m}} \{M_\iota :\ \iota \neq [k-1]_m\}$
        end
    end
    if $S = M_{[k+1]_m}$ then
        if $S:\ \mathsf{Vrf}(\sigma_{[k-1]_m})$ then
            $S:\ Y'_{[k+1]_m} = X_{[k+1]_m} - n_{[k+1]_m} Z_{[k-1]_m}$
            $S \xrightarrow{Y'_{[k+1]_m}} \{M_\iota :\ \iota \neq [k+1]_m\}$
        else
            $S:\ \sigma_{[k-1]_m} = \mathsf{Sign}(\{t+2\})$
            $S \xrightarrow{t+2, \sigma_{[k+1]_m}} \{M_\iota :\ \iota \neq [k+1]_m\}$
        end
    end
end
begin Key update
    $\mathsf{KeyDerivation}(Y_0, \ldots, Y'_{[k-1]_m}, Y'_{[k+1]_m}, \ldots, Y_{m-1})$
end
```

Java for ID-based key agreement

JIKA cryptographic framework implements the ID-based signature schemes and key agreement protocols described before. It simulates a key generation center (KGC) offering an ID-based key distribution service for all proposed algorithms. The JIKA implementation was tested on two different devices: *PC platform* and *Raspberry PI* [Upton and Halfacree, 2013]. Raspberry PI is a low-cost ARM-based device, become popular in the last years. It runs different Linux distributions and supports common programming languages (Java, C/C++, Python, Perl). Moreover, Raspberry PI allows network cards configuration in order to perform ad hoc communications. In this Chapter, we describe JIKA software architecture which relys on *jPair* [Dong, 2010] library. It implements mathematical operations required by arithmetic over elliptic curves and their embedded fields. Although others libraries exist with similar characteristics (e.g. the reference benchmark C library PBC of [Lynn, 2007] and its Java porting jPBC of [De Caro and Iovino, 2011]), jPair is a pure Java implementation with no dependencies on external libraries and a very small memory footprint. Java allows for portability and due to continuous improvements in the HotSpot technology, latest runtime environments greatly outperform previous versions, thus being practical also in network constrained environments. At the end of this Chapter, we discuss computing times of JIKA signature schemes and key agreement protocols.

6.1 JIKA software architecture

JIKA software architecture complies with modular programming, in which the functionality of a program is supported into independent and interchangeable modules. In this way, we can reuse modules to design new schemes and protocols with extended functionality. JIKA software architecture is depicted in Figure 6.1, it consists in the following modules:

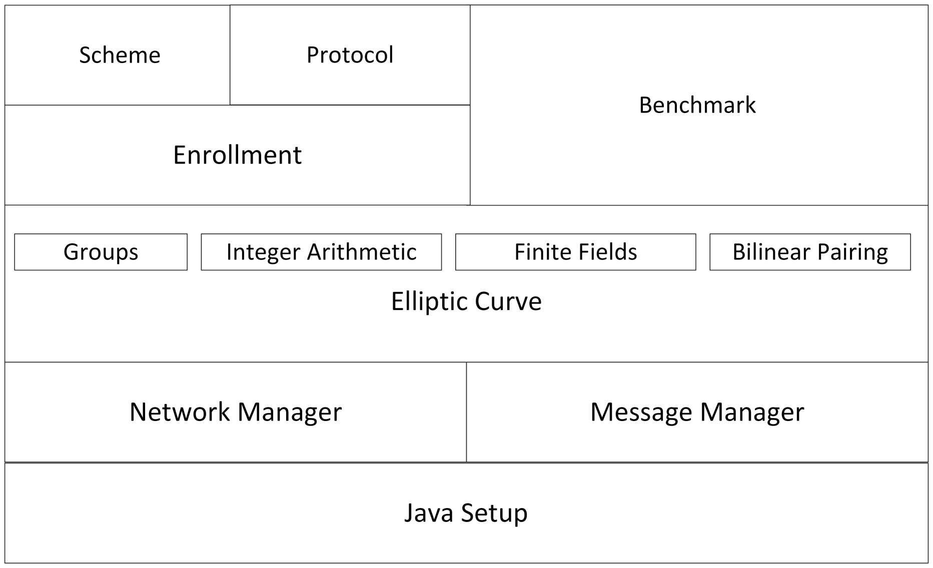


Fig. 6.1: JIKA framework software architecture.

- The *Java Setup* module relates to the Java virtual machine installed on the operating platforms. On both platforms we installed the Java Standard Edition Embedded 8. It introduces a new concept called Compact Profiles, which enables reduced memory footprint for applications that do not require the entire Java Standard Edition. This version supports three profiles: compact1, compact2 and compact3. Each Profile contains all of the APIs in the previous profiles, adding appropriate APIs on top. We used NetBeans 8 IDE for JIKA development, which supports compact Profiles and allows the deployment of Java application remotely.

- The *Network Manager* module is responsible of node synchronization, data transmission and error handling. It was implemented through the JGroups toolkit library [JGroups, 2014]. For any given TCP or UDP communication channel, JGroups offers services for joining and leaving, membership detection and notification, detection and removal of crashed users, sending and receiving of unicast and multicast messages. The most powerful feature of JGroups is its flexible protocol stack, which allows developers to adapt it to exactly match their application requirements and network characteristics. We chosen an UDP asynchronous communication channel that guarantee a better performance then TCP.

- A communication protocol requires that data are sent and received according to specific packeting rules (data field order and type). The *Message Manager* (MM) module is responsible of data serialization and de-serialization. These operations split and merge data sequencing due to special characters as separators. It is mandatory for MM to know order and type of data. For example, referring to eFG protocol, node A has to send the data fields A, R_A, T_A to recipient node B. The MM module sends an UDP packet which results of the serialization of three fields above. The MM module of recipient node B performs the inverse operations of de-serialization in order to properly identify the fields.

- The *Elliptic Curve* module implements mathematical operations required by elliptic curve module arithmetic. This includes finite fields, groups, integer arithmetic, bilinear pairings and hash functions. Many APIs of this module were implemented using jPair library [Dong, 2010] which was extended by adding the elliptic curves nssTate160, nssTate192 and nssTate224 (6.1).

- The module *Enrollment* allows nodes to store in memory cryptographic materials generated by the key generation center (KGC). The KGC performs enrollment off-line and then it distributes cryptographic materials on nodes through a secure out-of-band communication channel with authenticity and confidentiality properties.

- Module *Scheme* implements the ID-based signature schemes IBS-1, IBS-2 and GS-IBS. Each scheme requires the implementation of two methods: (Sign) for signature generation and (Vrf) for signature verification algorithms. The Sign method takes in input a message and the signer's secret key, and returns the related message signature. The Vrf method takes in input the message, its signature and node identity, and returns a boolean value which indicates if verification succeeds or not.

- Finally, the *Protocol* module implements the ID-based key agreement protocols eFG, GKA v1 and GKA v2. In particular, GKA protocol comes in three different flavours protocols, which depends on the specific signature scheme adopted for message authentication.

6.2 Implementation of elliptic curves and tate pairing

JIKA extends jPair implementation [Dong, 2010] by adding some elliptic curves. jPair natively supports some supersingular and ordinary elliptic curves over large prime fields, and implements all the arithmetic required for finite fields and tate pairings over such curves. In general, bilinear pairing computations are quite expensive and require carefully optimized implementations. The implementation of tate pairings in jPair is inspired to [Chatterjee et al., 2005], that introduced the idea of *encapsulated point doubling/addition* based on projective coordinate system. Many different coordinate systems can used to represent points on elliptic curves. The selection of the points coordinate systems has a significant influence on the performance of the elliptic curve arithmetic operations. For instance, point multiplication algorithm can be expedited by using efficient representation of elliptic curve points using projective coordinates. In general Jacobian coordinates are faster in doubling algorithm then projective coordinates and slower in addition algorithm. This approach applies to elliptic curves over large prime fields with embedding degree 2, leading to an improvement of around 33% over the best known algorithm [Izu and Takagi, 2003] for the general case where the curve parameter a of equation (3.2) for $p > 3$ is an arbitrary field element, and around 20% over the algorithm in [Scott, 2005] in the special case where $a = -3$. Natively, jPair supports only the following curves over prime fields $\mathbb{F}_p$, where $p \equiv 3 \mod 4$:

- supersingular curves of equation $y^2 = x^3 + x$, for randomly chosen values of the prime p and of the prime order l of group $\mathbb{G}$, where l divides $p + 1$ (3).

- ordinary curves of equation $y^2 = x^3 - 3x + b$, for some selected values of the 160-bit prime l, and some corresponding values for the prime p and the integer b.

We have extended the jPair API by adding to the second class curves for selected values of 192 and 224-bit primes l. Since all the above curves have embedding degree $k = 2$, one has to choose suitable big cofactors c in order to have an adequate level of security (see details in Section 3). For example, to achieve security equivalent to 1024-bit RSA, l should be at least 160-bit and p should be at least 512-bit. Analogously, to achieve 2048-bit RSA security, l should be 224-bit and p should be 1024-bit. Table 6.1 shows the bilinear pairing implemented at the time being in our extension of jPair, along with the order (in bit size) of group $\mathbb{G}$, (and $\mathbb{G}'$ for ordinary curves), and the corresponding order (in bit size) of $\mathbb{F}_p$ such that the resulting bilinear DL system has a consistent level of security.

Name	Elliptic curve	Order of $\mathbb{G}$	Order of $\mathbb{F}_p$
ssTate160	$y^2 = x^3 + x$	160-bit	512-bit
ssTate192	$y^2 = x^3 + x$	192-bit	768-bit
ssTate224	$y^2 = x^3 + x$	224-bit	1024-bit
nssTate160	$y^2 = x^3 - 3x + b$	160-bit	512-bit
nssTate192	$y^2 = x^3 - 3x + b$	192-bit	768-bit
nssTate224	$y^2 = x^3 - 3x + b$	224-bit	1024-bit

Tab. 6.1: The Elliptic Curves implemented in JIKA and used in our tests. Names are consistent with those of jPair. They indicate: the type of curve (supersingular (ss) or ordinary (nss)), type of pairing supported (Tate) and the order in bit size of the cyclic group(s) considered on the curve.

6.3 Performance evaluation

A set of tests were executed on a PC platform and a Raspberry PI, in order to measure the performances of schemes and protocols implemented in JIKA. We used an Intel Core 2 Duo E6600 with 4096 MBytes of RAM, equipped with Microsoft Windows 7 SP1 and Java Standard Edition 8. The Raspberry PI is an embedded device developed in UK by the Raspberry Pi Foundation. Its design is based on a Broadcom BCM2835 system on chip (SoC), which includes an ARM1176JZF-S 700MHz processor, 512 Megabytes of RAM and an SD card. We installed Java Standard Edition Embedded 8 on Raspberry PI. The experiment results are the average of 1000 executions, in order to keep out computing time outliers, due to the Java virtual machine and its interactions with operating system.

6.3.1 Enrollment

The first set of tests measures enrollment computing times. As we told in Section 3.4, Enrollment consists in the initialization of public parameters and generation of the user signature keys for schemes IBS-1, IBS-2 and GS-IBS. Moreover, the enrollment includes the pre-computation of the bilinear pairing value $e(P, P')$ required for signature scheme IBS-2.

PC platform		
	Computing Time (s)	
Procedure Enrollment	EC. ssTate160	EC. nssTate160
$n = 10$	0.96	0.767
$n = 50$	3.893	3.427
$n = 100$	7.392	6.930
Raspberry PI		
	Computing Time (s)	
Procedure Enrollment	EC. ssTate160	EC. nssTate160
$n = 10$	34.795	31.288
$n = 50$	148.858	141.645
$n = 100$	294.137	277.206

Tab. 6.2: Computing times for the enrollment of 10, 50 and 100 nodes using elliptic curves ssTate160 and nssTate160 on the PC platform and a Raspberry PI.

Table 6.2 shows computing times on the PC and Raspberry PI for the enrollment of n nodes (m = 10, 50, 100). On both operating platforms and for each of the indicated elliptic curves, the results show a linear increment of computing times versus the number of nodes. However, results are 5-15% better on nssTate160 curve than on ssTate160. Moreover, the test results indicate that the enrollment procedure runs on the PC platform 30-40 times faster than on a Raspberry PI. This is a remarkable gap in performance, due to the constrained computing resource of the embedded devices. However, enrollment is a one-time procedure that can be executed off-line on any Java equipped platform.

6.3.2 Signature schemes

Table 6.3 shows the computing times required by signing and verification algorithms of IBS-1, IBS-2 and GS-IBS schemes. Overall the three IBS schemes perform very well, with computing times never exceeding the threshold of 100 milliseconds on the PC platform and 2 seconds on the Raspberry PI. IBS-2 is the most efficient scheme: it slightly outperforms IBS-1 of about 8% in both signing and verification time, but its gain versus GS-IBS is about 45-50% in signing time and 15-20% in verification time, where percentages are higher on the supersingular curve (since computations are slightly faster here than on the ordinary curve).

PC platform		
	Computing Time (ms)	
Scheme	EC. ssTate160	EC. nssTate160
	Signature	
IBS-1 Sgn	38	42
IBS-2 Sgn	35	39
GS-IBS Sgn	69	74
	Verification	
IBS-1 Vrf	74	79
IBS-2 Vrf	69	74
GS-IBS Vrf	85	89

Tab. 6.3: Computing times required by IBS-1, IBS-2, and GS-IBS signature and verification algorithms on the PC platform.

Raspberry PI		
	Computing Time (ms)	
Scheme	EC. ssTate160	EC. nssTate160
	Signature	
IBS-1 Sgn	661	736
IBS-2 Sgn	631	702
GS-IBS Sgn	1491	1566
	Verification	
IBS-1 Vrf	1493	1534
IBS-2 Vrf	1425	1462
GS-IBS Vrf	1752	1798

Tab. 6.4: Computing times required by IBS-1, IBS-2, and GS-IBS signature and verification algorithms on a Raspberry PI.

6.3.3 *Key agreement protocols*

We discuss a set of tests relative to the key agreement protocols introduced in the previous Chapter. In some cases, we consider simulations instead of runs over real networks, in order to keep out initialization and synchronization time delays, that can introduce a substantial noise in performance measurement.

eFG protocol

Table 6.5 show computing times of the eFG protocol versus the two-party key agreement protocol ECMQVS [Rogaway et al., 2001]. This last protocol is widely standardized and it is considered one of the most efficient two-party key agreement protocol. It requires four scalar point multiplication and three hash function computations for each node. However, unlike the eFG protocol, ECMQVS requires Public Key Certificates (PKCs) for nodes authentication. It's difficult to estimate the costs of PKCs management in term of computing time, but anyway it results in a significant overhead if compared with the ID-based approach.

PC platform		
	Computing Time (ms)	
Elliptic curve	eFG	ECMQVS
nssTate160	99	135
nssTate192	105	138
nssTate224	108	140
Raspberry PI		
	Computing Time (ms)	
Elliptic curve	eFG	ECMQVS
nssTate160	2088	3105
nssTate192	2280	3267
nssTate224	2439	3490

Tab. 6.5: Computing times for eFG and ECMQVS protocols simulations, using elliptic curves nssTate160, nssTate192, nssTate224 on the PC platform and a Raspberry PI.

A comparison in computing times between eFG and ECMQVS protocol simulations, illustrated in Table 6.5, shows that eFG is a very efficient protocol. Indeed, it outperforms ECMQVS of 25% on the PC platform and of 30% on a Raspberry PI. A second set of tests was relative to executions of eFG on a real network and included times required for message exchanges. The tests were performed using both wired IEEE 802.3 network card and wireless IEEE 802.11 adapter. Table 6.6 reports averages computing times on both platforms over elliptic curves nssTate160, nssTate192 and nssTate224. The values indicate that eFG is about 20 times faster on the PC platform than Raspberry PI. Finally, Table 6.7 reports computing times of protocol eFG including nodes initialization, messages exchange and synchronizations time delays. Also in this case, execution times on the PC platform were about

20 times faster than on the Raspberry PI.

PC platform		
	Computing Time (ms)	
Elliptic curve	Wired	Wireless
nssTate160	140	161
nssTate192	151	171
nssTate224	155	180
Raspberry PI		
	Computing Time (ms)	
Elliptic curve	Wired	Wireless
nssTate160	2450	2528
nssTate192	2780	2849
nssTate224	2970	3042

Tab. 6.6: Computing times of protocol eFG on a real network using the elliptic curves nssTate160, nssTate192 and nssTate224. The values include message exchange times.

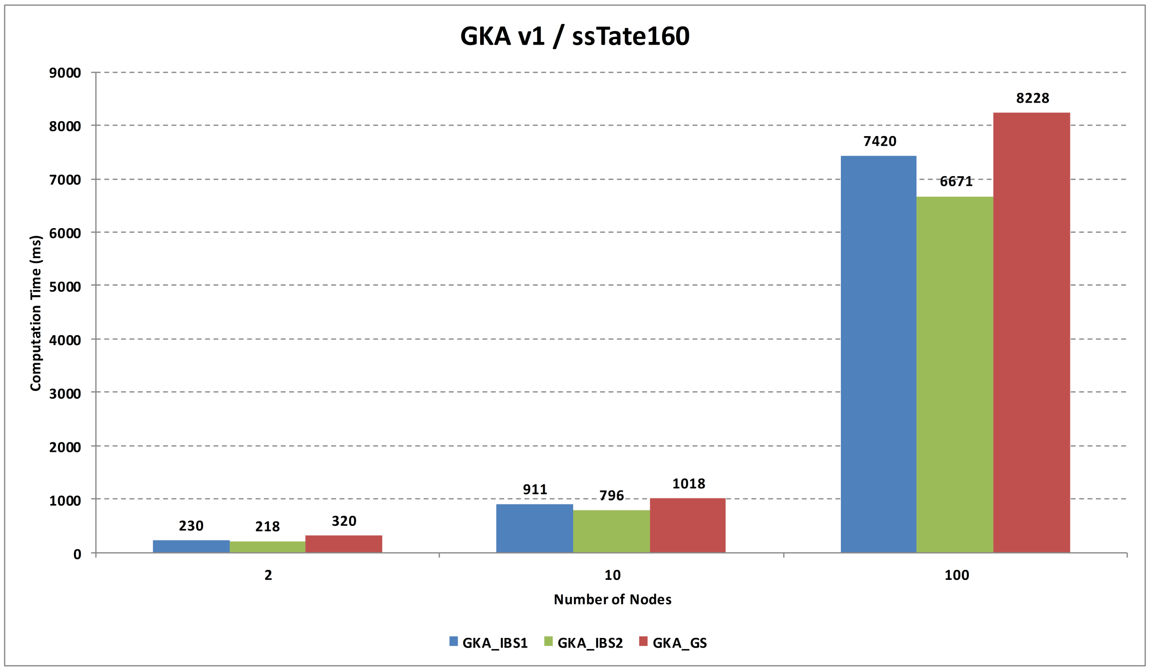
PC platform		
	Computing Time (ms)	
Elliptic Curve	Wired	Wireless
nssTate160	232	290
nssTate192	259	333
nssTate224	295	376
Raspberry PI		
	Computing Time (ms)	
Elliptic Curve	Wired	Wireless
nssTate160	3002	3060
nssTate192	3250	3302
nssTate224	3384	3471

Tab. 6.7: Computing times of protocol eFG on a real network using the elliptic curves nssTate160, nssTate192 and nssTate224. The values include nodes initialization, messages exchange and synchronizations time.

GKA protocols

In this Section, we discuss a set of tests relative to GKA protocols. Specifically, GKA v1 and GKA v2, were combined with the signature schemes IBS-1,IBS-2 and GS-IBS in order to compare their performances.

The experiment results are relative to elliptic curves *ssTate160* and *nssTate160*. Figures 6.2 and 6.3 show total computing times required on the PC platform to derive a group session key versus number of nodes involved. The same tests were repeated on a Raspberry PI, and the experiment results are reported in Figures 6.4 and 6.5. Besides the fact that the better performance of schemes IBS-1 and IBS-2 can lead to relevant improvements for a large number of nodes, the main point to notice here is that one has to replace the GKA v1 protocol with protocol GKA v2 in order to get group key agreement with a quite large group of users at an affordable computing time. This is because protocol GKA v1 does a number of signatures and verifications which linearly increases with the number of the nodes involved, whilst GKA v2 only performs a constant number of such verifications. The best performances on both Raspberry PI and PC platform were reached by GKA v2 with signature scheme IBS-2. The computing times on Personal Computer were about 30-40 times faster than on a Raspberry PI.



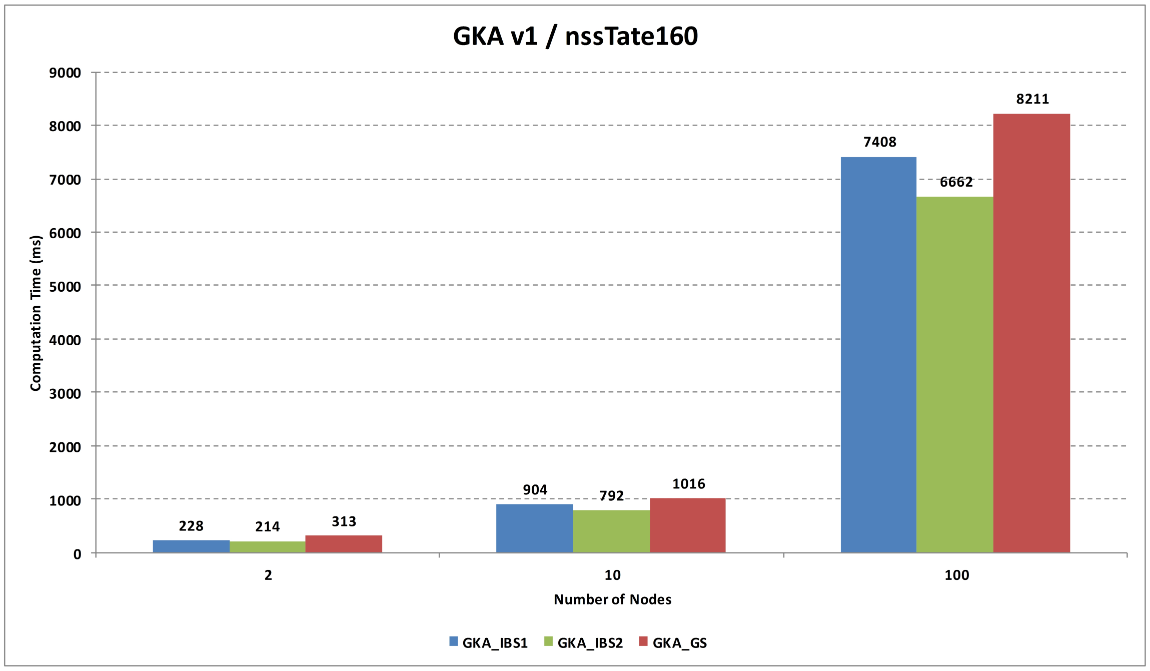
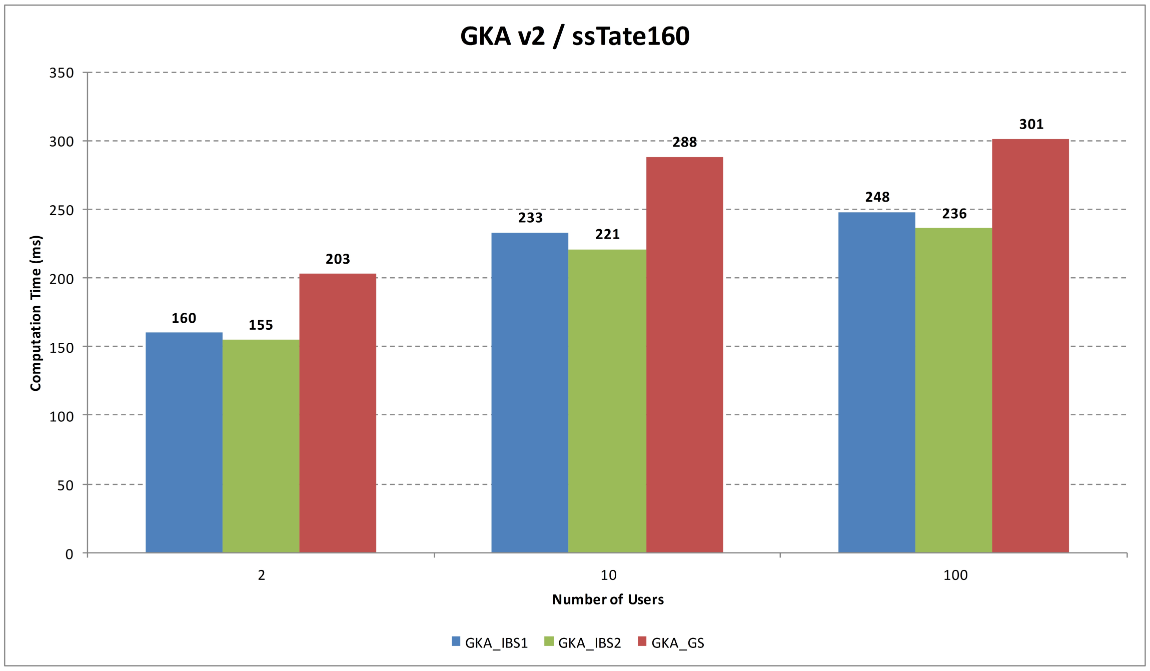


Fig. 6.2: GKA v1 computing times versus involved parties on the PC platform using elliptic curves ssTate160 and nssTate160, respectively.



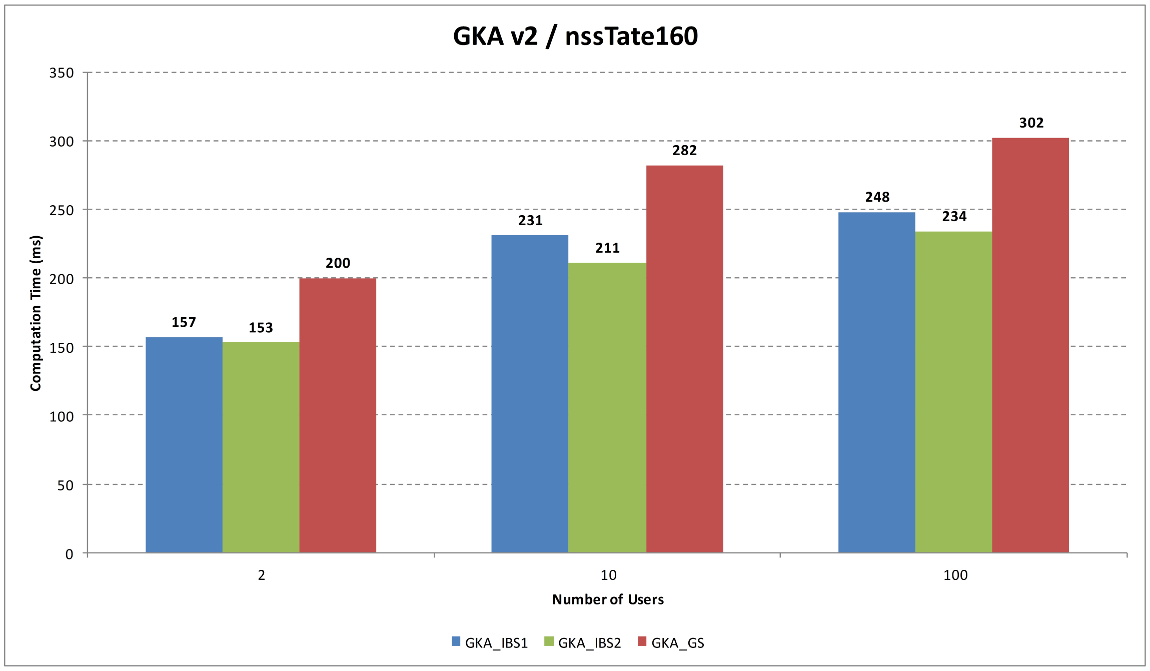
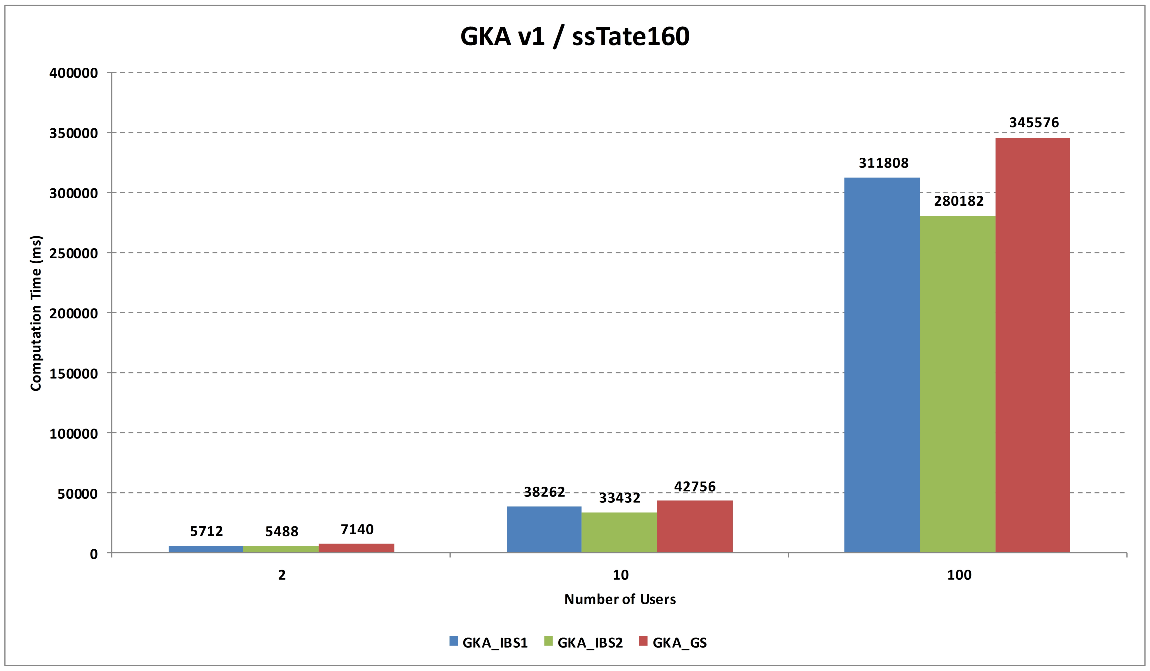


Fig. 6.3: GKA v2 computing times versus involved parties on the PC platform using elliptic curves ssTate160 and nssTate160, respectively.



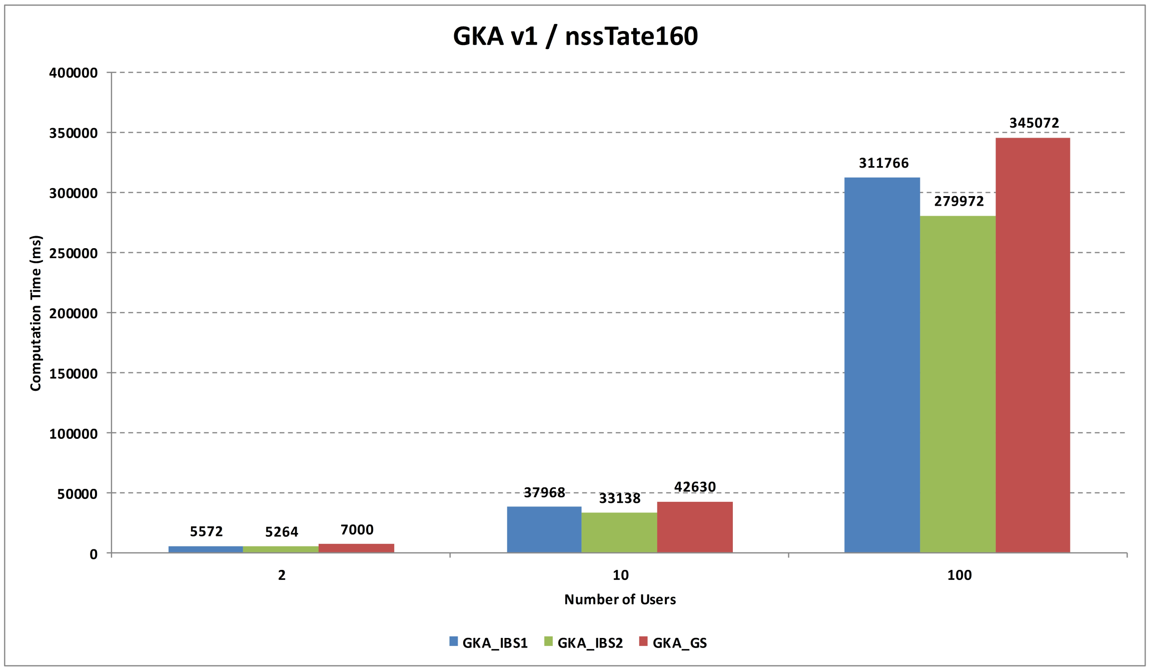
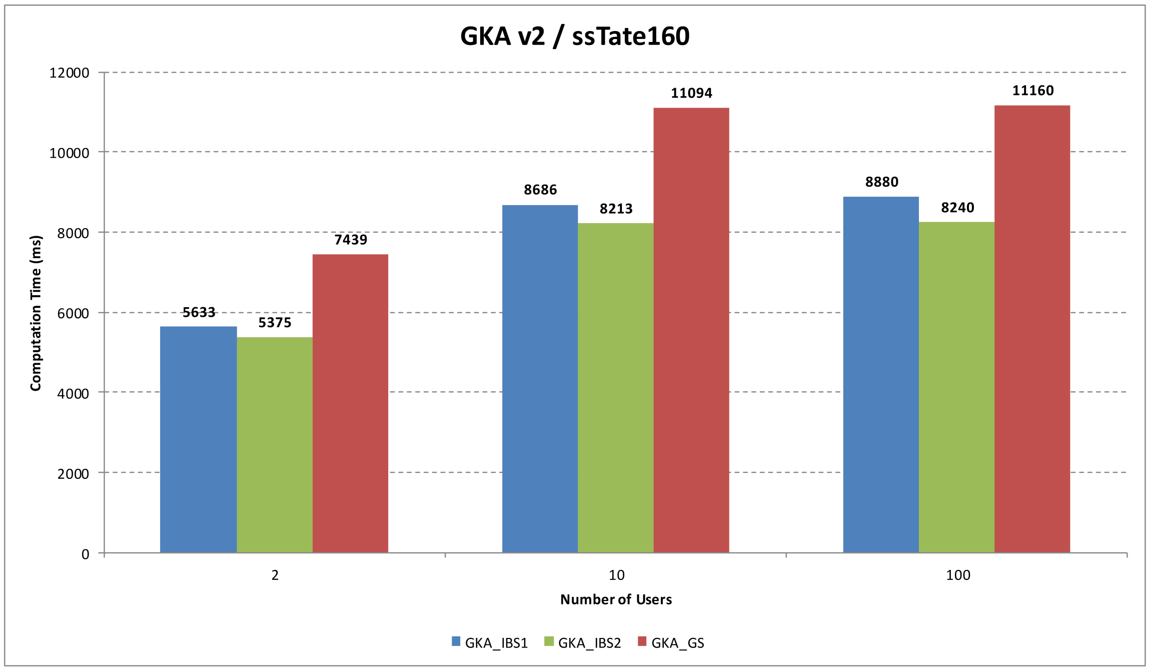


Fig. 6.4: GKA v1 computing times versus involved parties on Raspberry PI using elliptic curves ssTate160 and nssTate160, respectively.



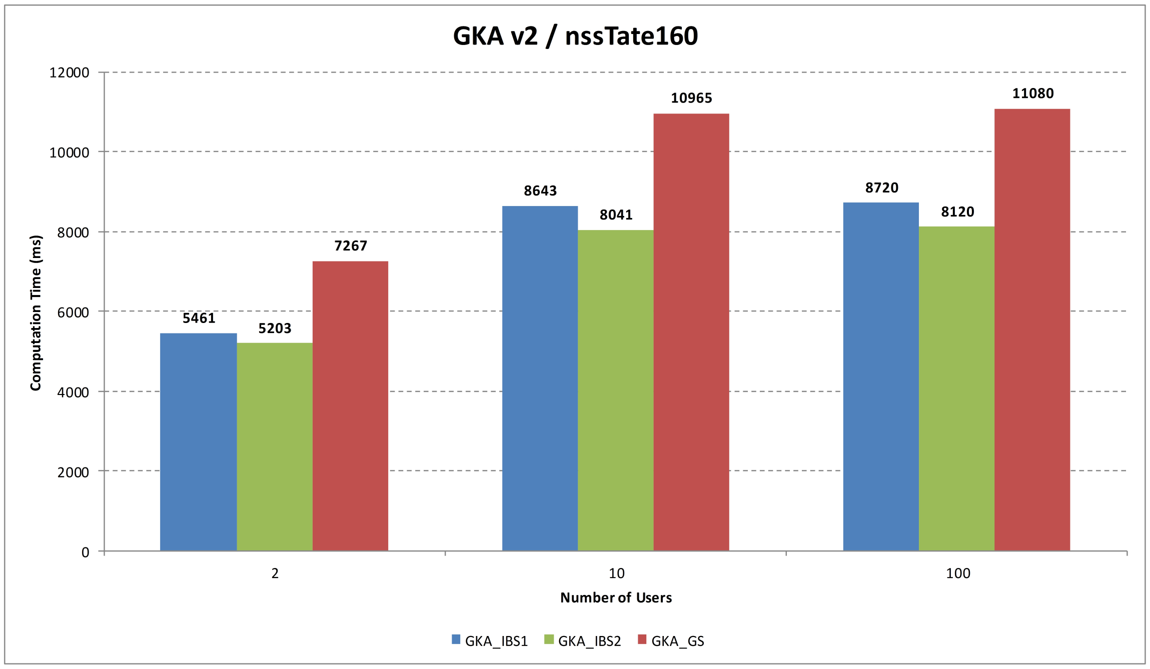


Fig. 6.5: GKA v2 computing times versus involved parties on Raspberry PI using elliptic curves ssTate160 and nssTate160, respectively.

Finally, we compared the following four group key agreement protocols with respect to involved parties: the unauthenticated protocol of [Burmester and Desmedt,

1994] over standard groups (BD protocol), the unauthenticated protocol derived by GKA (uGKA), GKA v2/IBS-2 and GKA v2/ECDSA. It is worth to stress here that all the above protocols use two communication rounds and the same number of messages to compute the key.
This set of tests was performed to evaluate both the gain in performance due to elliptic curve cryptography and the overhead introduced by pairing-based signature versus standard signatures.

Personal Computer				
	Computing Time (ms)			
Number of users	BD	uGKA	GKA v2/IBS-2	GKA v2/ECDSA
$m = 10$	143	24	196	158
$m = 30$	509	29	197	160
$m = 50$	2886	35	201	162
$m = 70$	10443	38	204	166
$m = 90$	25881	41	207	169
Raspberry PI				
	Computing Time (ms)			
Number of users	BD	uGKA	GKA v2/IBS-2	GKA v2/ECDSA
$m = 10$	34425	1063	8017	6459
$m = 30$	/	1143	8027	6505
$m = 50$	/	1218	8084	6580
$m = 70$	/	1289	8163	6658
$m = 90$	/	1373	8261	6746

Tab. 6.8: Computing times of protocol DB and protocols uGKA, GKA/IBS-2 and GKA/ECDSA over the elliptic curve ssTate160 results are relative to the PC platform and the Raspberry PI, respectively.

The advantages in using elliptic curve cryptography over standard cryptography are clearly shown by the comparison between protocols BD and uGKA. Notice

however that a significant amount of the gap in performance is due to the different way to compute the session key. As we told in Section 5.3, indeed, protocol BD requires a total of $m(m+1)$ modular exponentiations, whilst uGKA requires just one scalar point multiplication (corresponding to a full modular exponentiation) and $2m-2$ point additions (corresponding to $2m-2$ full modular exponentiations). In particular, the large amount of memory and computational power required by modular exponentiations hanged executions of protocol BD for more than $m = 90$ users on the PC platform and $m \geq 30$ users on the Raspberry PI.
Ultimately the obtained results show the effectiveness of GKA v2 for large group of parties also on constrained devices such as the Raspberry PI. Moreover, the overhead introduced for this protocol by pairing-based computations through scheme IBS-2 was about 20% of the computing time of the scheme ECDSA. Although that gap is not negligible, we have to consider that ECDSA requires the management of PKCs and a more expensive initialization of nodes, since the lack of identities attributes doesn't allow an easy arrangement of nodes into a cyclic lexicographic order.

Conclusions

WANETs applications are quickly increasing over time and actually, due to unique features (such as node mobility and self-organization), they are often preferred to infrastructure-based wireless networks. However, the open nature of the wireless communication channel exposes WANETs to a great number of security threats. In this context, the aim of this thesis was to propose a cryptographic framework, named JIKA, which provides authentication through an ID-based key distribution mechanism and key agreement protocols.

In particular, JIKA includes a two-party key agreement protocol (eFG) and two new group key agreement protocols (GKA v1 and GKA v2). We coupled the group key agreement protocols with the new ID-based signature schemes (IBS-1 and IBS-2), in order to get authenticated group key agreement protocols that require low computational costs and bandwidth for each of the involved parties.

We provided a proof sketch for all proposed algorithms and contextually demonstrated their correctness. Furthermore, we scheduled a set of benchmarks over Raspberry PI and PC operating platforms to evaluate the performances of JIKA algorithms in term of computing times, comparing them with performances of others standard algorithms. Interestingly, our results obtained by eFG protocol which outperformed ECMQVS of with a percentile degree of 25% on the PC platform and of 30% on a Raspberry PI. Moreover, a very good performances on both Raspberry PI and PC platforms were reached by GKA v2 with signature scheme IBS-2.

The results of this thesis pave the way for future work/s planned to extend JIKA with others algorithms and contextually to increase its performances with the implementation of new pairing-friendly curves and cryptographic primitives for arithmetic over elliptic curves. Moreover, we will consider the implementation of JIKA with others programming languages spreading its applicability to new generation platforms.

Bibliography

Augot, D., Bhaskar, R., Issarny, V., and Sacchetti, D. (2007). A three round authenticated group key agreement protocol for ad hoc networks. *Pervasive and Mobile Computing*, 3(1):36–52.

Barreto, P. S., Libert, B., McCullagh, N., and Quisquater, J.-J. (2005). Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *Advances in Cryptology - ASIACRYPT 2005*, pages 515–532. Springer.

Becker, K. and Wille, U. (1998). Communication complexity of group key distribution. In *Proceedings of the 5th ACM conference on Computer and communications security*, pages 1–6. ACM.

Bellare, M., Canetti, R., and Krawczyk, H. (1998). A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 419–428. ACM.

Bellare, M. and Rogaway, P. (1993). Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM.

Berg, J. S. (2008). *Broadcasting on the short waves, 1945 to today.* McFarland.

Boneh, D. and Franklin, M. (2001). Identity-based encryption from the weil pairing. In *Advances in Cryptology CRYPTO 2001*, pages 213–229. Springer.

Boneh, D., Lynn, B., and Shacham, H. (2004). Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319.

Boyd, C. and Mathuria, A. (2003). *Protocols for authentication and key establishment*. Springer.

Boyd, C. and Nieto, J. M. G. (2002). Round-optimal contributory conference key agreement. In *Public Key Cryptography - PKC 2003*, pages 161–174. Springer.

Bresson, E. and Catalano, D. (2004). Constant round authenticated group key agreement via distributed computation. In *Public Key Cryptography–PKC 2004*, pages 115–129. Springer.

Bresson, E., Chevassut, O., and Pointcheval, D. (2001a). Provably authenticated group diffie-hellman key exchange in the dynamic case. In *Advances in Cryptology - ASIACRYPT 2001*, pages 290–309. Springer.

Bresson, E., Chevassut, O., and Pointcheval, D. (2002). Dynamic group diffie-hellman key exchange under standard assumptions. In *Advances in Cryptology - EUROCRYPT 2002*, pages 321–336. Springer.

Bresson, E., Chevassut, O., Pointcheval, D., and Quisquater, J.-J. (2001b). Provably authenticated group diffie-hellman key exchange. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 255–264. ACM.

Burmester, M. and Desmedt, Y. (1994). A secure and efficient conference key distribution system. In *Pre-proceedings EUROCRYPT'94*, pages 275–286. Scuola Superiore G. Reiss Romoli.

Burmester, M. and Desmedt, Y. (1995). A secure and efficient conference key distribution system. In Santis, A., editor, *Advances in Cryptology - EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer Berlin Heidelberg.

Burmester, M. and Desmedt, Y. (2005). A secure and scalable group key exchange system. *Information Processing Letters*, 94(3):137–143.

Chatterjee, S., Sarkar, P., and Barua, R. (2005). Efficient computation of tate pairing in projective coordinate over general characteristic fields. In *Information Security and Cryptology ICISC 2004*, pages 168–181. Springer.

Cocks, C. (2001). An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding*, pages 360–363. Springer.

Coppersmith, D. (1984). Fast evaluation of logarithms in fields of characteristic two. *Information Theory, IEEE Transactions on*, 30(4):587–594.

De Caro, A. and Iovino, V. (2011). jpbc: Java pairing based cryptography. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 850–855. IEEE.

Debian (2015). WiFi Ad-hoc Network. https://wiki.debian.org/WiFi/AdHoc.

Desmedt, Y. (2011). Man-in-the-middle attack. In *Encyclopedia of Cryptography and Security*, pages 759–759. Springer.

Di Pietro, R., Guarino, S., Verde, N., and Domingo-Ferrer, J. (2014). Security in wireless ad-hoc networks–a survey. *Computer Communications*, 51:1–20.

Diem, C. (2011). On the discrete logarithm problem in elliptic curves. *Compositio Mathematica*, 147(01):75–104.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654.

Diffie, W., Van Oorschot, P. C., and Wiener, M. J. (1992). Authentication and authenticated key exchanges. *Designs, Codes and cryptography*, 2(2):107–125.

Dong, C. (2010). Jpair: A quick introduction. https://personal.cis.strath.ac.uk/changyu.dong/jpair/intro.html.

Dutta, R. and Barua, R. (2005). Constant round dynamic group key agreement. In *Information Security*, pages 74–88. Springer.

Dutta, R. and Barua, R. (2008). Provably secure constant round contributory group key agreement in dynamic setting. *Information Theory, IEEE Transactions on*, 54(5):2007–2025.

Dutta, R., Barua, R., and Sarkar, P. (2004). Provably secure authenticated tree based group key agreement. In *Information and Communications Security*, pages 92–104. Springer.

Eschenauer, L. and Gligor, V. D. (2002). A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM.

Fiore, D. and Gennaro, R. (2010). Identity-based key exchange protocols without pairings. In *Transactions on computational science X*, pages 42–77. Springer.

Frey, G., Muller, M., and Ruck, H.-G. (1999). The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *Information Theory, IEEE Transactions on*, 45(5):1717–1719.

Gentry, C. and Silverberg, A. (2002). Hierarchical id-based cryptography. In *Advances in Cryptology - ASIACRYPT 2002*, pages 548–566. Springer.

Gislason, D. (2008). *ZigBee wireless networking*. Newnes.

Goldwasser, S. and Micali, S. (1982). Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377. ACM.

Goldwasser, S., Micali, S., and Rivest, R. L. (1988). A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308.

Hess, F. (2003). Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography*, pages 310–324. Springer.

Ingemarsson, I., Tang, D., and Wong, C. (1982). A conference key distribution system. *Information Theory, IEEE Transactions on*, 28(5):714–720.

Izu, T. and Takagi, T. (2003). Efficient computations of the tate pairing for the large mov degrees. In *Information Security and Cryptology ICISC 2002*, pages 283–297. Springer.

JGroups (2014). Jgroups toolkit. http://www.jgroups.org/.

Johnson, D., Menezes, A., and Vanstone, S. (2001). The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63.

Joye, M. and Neven, G. (2009). *Identity-based cryptography*, volume 2. IOS Press.

Katz, J. and Yung, M. (2003). Scalable protocols for authenticated group key exchange. In *Advances in Cryptology - CRYPTO 2003*, pages 110–125. Springer.

Katz, J. and Yung, M. (2007). Scalable protocols for authenticated group key exchange. *Journal of Cryptology*, 20(1):85–113.

Koblitz, N. (1994). *A course in number theory and cryptography*, volume 114. Springer.

Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J., and Yoo, S. (2004). Secure key issuing in id-based cryptography. In *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation-Volume 32*, pages 69–74. Australian Computer Society, Inc.

Lynn, B. (2002). Authenticated identity-based encryption. *IACR Cryptology ePrint Archive*, 2002:72.

Lynn, B. (2007). *On the implementation of pairing-based cryptosystems*. PhD thesis, Stanford University.

Menezes, A. J., Okamoto, T., and Vanstone, S. A. (1993). Reducing elliptic curve logarithms to logarithms in a finite field. *Information Theory, IEEE Transactions on*, 39(5):1639–1646.

Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

Mpitziopoulos, A., Gavalas, D., Konstantopoulos, C., and Pantziou, G. (2009). A survey on jamming attacks and countermeasures in wsns. *Communications Surveys & Tutorials, IEEE*, 11(4):42–56.

Nam, J., Lee, J., Kim, S., and Won, D. (2004). Ddh based group key agreement for mobile computing. *Cryptology e-Print Archive, Report*, 127.

Pathan, A.-S. K. (2010). *Security of self-organizing networks: MANET, WSN, WMN, VANET*. CRC press.

Pickholtz, R. L., Schilling, D. L., and Milstein, L. B. (1982). Theory of spread-spectrum communications–a tutorial. *Communications, IEEE Transactions on*, 30(5):855–884.

Pollard, J. (1978). Monte carlo methods for index computation (modp). *Mathematics of Computation*, pages 918–924.

Rogaway, P., Bellare, M., and Boneh, D. (2001). Ecmqvs (from sec 1).

Rossi, F. and Schmid, G. (2015). Implementing identity-based key agreement in embedded devices. In *PECCS 2015 - Proceedings of the 5st International Conference* *on Pervasive and Embedded Computing and Communication Systems, Angers, France, 11-13 February, 2015*, pages 117–123.

Sakai, R. and Kasahara, M. (2003). Id-based cryptosystems with pairing on elliptic curve. *IACR Cryptology ePrint Archive*, 2003:54.

Sarkar, S. K., Basavaraju, T., and Puttamadappa, C. (2007). *Ad hoc mobile wireless networks: principles, protocols and applications.* CRC Press.

Schirokauer, O. (2000). Using number fields to compute logarithms in finite fields. *Mathematics of Computation of the American Mathematical Society*, 69(231):1267–1283.

Schmid, G. and Rossi, F. (2011). Secure ad-hoc routing through a-codes. In *PECCS 2011 - Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems, Vilamoura, Algarve, Portugal, 5-7 March, 2011*, pages 151–156.

Schmid, G. and Rossi, F. (2012). QR code-based identification with mobile devices. In *PECCS 2012 - Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems, Rome, Italy, 24-26 February, 2012*, pages 79–86.

Schnorr, C.-P. (1991). Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174.

Scott, M. (2005). Computing the tate pairing. In *Topics in Cryptology CT-RSA 2005*, pages 293–304. Springer.

Shamir, A. (1985). Identity-based cryptosystems and signature schemes. In *Advances in cryptology*, pages 47–53. Springer.

Silverman, J. H. (2009). *The arithmetic of elliptic curves*, volume 106. Springer.

Standards association (2005). IEEE 802.15: Wireless personal area networks (PANs). http://standards.ieee.org/about/get/802/802.15.html.

Standards association (2012). IEEE 802.11: Wireless local area networks (LANs). http://standards.ieee.org/about/get/802/802.11.html.

Steiner, M., Tsudik, G., and Waidner, M. (1996). Diffie-hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM conference on Computer and communications security*, pages 31–37. ACM.

Upton, E. and Halfacree, G. (2013). *Raspberry Pi User Guide*. John Wiley & Sons.

Yao, G., Wang, H., and Jiang, Q. (2008). An authenticated 3-round identity-based group key agreement protocol. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 538–543. IEEE.

Zhang, F., Safavi-Naini, R., and Susilo, W. (2004). An efficient signature scheme from bilinear pairings and its applications. In *Public Key Cryptography–PKC 2004*, pages 277–290. Springer.