MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Key Agreement for Mobile Phones

MASTER'S THESIS

Bc. Shkodran Gerguri

Brno, Spring 2011

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Advisor: RNDr. Jan Bouda, Ph.D.

Acknowledgement

I would like to thank my advisor RNDr. Jan Bouda, Ph.D., for his valuable advice and suggestions.

My endless gratitude belongs to my mother for creating an environment that made this thesis possible, despite all the circumstances.

Contents

1 **Introduction** . . . 2
2 **Key Establishment: Taxonomy and Definitions** . . . 4
2.1 *Key Agreement in Groups* . . . 5
2.2 *Attacker Model and Security Assumptions* . . . 8
2.3 *Proofs of Security* . . . 9
3 **Group Key Agreement Protocols** . . . 12
3.1 *Protocols on Rings* . . . 14
3.1.1 Ingemarsson et al. . . . 14
3.1.2 Burmester-Desmedt . . . 16
3.1.3 Katz-Yung . . . 16
3.1.4 Dutta-Barua . . . 18
3.1.5 Generic n-party Diffie-Hellman – GDH.1-3 protocols 21
3.1.6 Authenticated Extensions of GDH . . . 25
3.2 *Tree-based protocols* . . . 26
3.2.1 (N)AGKA . . . 26
3.2.2 STR . . . 27
3.3 *Star-based Protocols* . . . 30
3.3.1 Boyd-Nieto . . . 30
3.3.2 Bresson et al. . . . 31
3.3.3 Augot et al. . . . 33
3.4 *Comparison* . . . 34
4 **Implementation** . . . 37
4.1 *Mobile Phone Platforms* . . . 38
4.2 *Communication Channels* . . . 41
4.3 *Protocol Selection* . . . 43
4.4 *Protocol Modifications* . . . 45
4.4.1 Message Format Modification . . . 47
4.5 *Implementation details* . . . 48
4.5.1 Member Run . . . 49
4.5.2 Leader Run . . . 50
5 **Conclusion** . . . 51

Chapter 1

Introduction

Consider the case of *Group Key Agreement* protocols – communication protocols that allow a set of n participants to agree upon a shared secret value over an insecure channel in the presence of passive and active adversaries. Secret values established in this way can be used as a secret key to enable confidential multicast comunication inside a group.

Using a mobile phone, these protocols can also be used as a "bootstrapping" device to agree on the secret key while the users are in close proximity of each other, with the key then being used on another device or computer. Utilizing wireless networking such as Bluetooth, the users can directly connect the devices to each other, potentially minimizing the amount of attackers that can eavesdrop or modify the traffic while providing the means to visually verify the computed values in the group.

The aim of this thesis is to provide an overview of group key agreement protocols that can be reasonably implemented in a mobile phone environment along with an implementation of a suitable protocol. Our focus is on protocols with operations in finite (cyclic) groups, as these can be implemented using modular arithmetics in groups of integers modulo a certain prime.

Chapter 2 serves as a general introduction to the area – we establish some of the concepts and terms associated with key agreement in groups, briefly discuss the related concept of key distribution, and discuss the security assumptions and a common security model used in formally proving that a given GKA protocol is secure.

The main text is contained in Chapters 3 and 4. Chapter 3 provides an overview of relevant group key agreement protocols, grouped together based on the logical hierarchy assumed in the group.

Discussion related to the practical implementation of the chosen protocols is then given in Chapter 4. We briefly discuss available implementation environments and the communication channels, providing rationale for our particular choice of the two for implementation. These are then used as a basis for selection of a suitable protocol. We also make the case for alterna-

tive message authentication mechanisms, as we discuss the difficulties of efficiently using digital signatures on messages using public key cryptography in our particular scenario. The rest of the chapter then deals with the overall architecture of implementation of the selected protocols and some of the implementation specifics that required a workaround.

Chapter 2

Key Establishment: Taxonomy and Definitions

Consider the following scenario – a group of users wants to establish a cryptographic key to communicate together in a confidential manner such that every participant from within the group can encrypt and decrypt messages sent within the group, but no outsider is capable of doing so. We refer to methods that allow the group of users to establish the secret key as the *group key establishment methods*. We can distinguish two types of key establishment methods depending on the amount of contribution every member of the group makes towards the shared secret value – key distribution and key agreement.

Key distribution refers to methods where a designated party, or a subset of designated parties, chooses the keying material and distributes it to the rest of the group; such parties may or may not be members of the group. Group key distribution can be built naturally using key distribution centers (KDC) – trusted third parties that share pairwise keys with the group members. In such scenario, the KDC generates the group key itself and uses the pairwise keys to send the encrypted key to the individual members of the group.

Key agreement methods allow all members of the group to contribute their own secret value towards the shared secret. Key agreement does not involve a trusted third party, and the members thus have to do all the computation themselves. Key agreement methods enjoy several advantages over key distribution methods, one of those being the *contributory* property, i.e., that all active participants in the protocol contribute to the resulting secret value, but without the help of an outside party.

A related notion to key distribution is that of *key management*. These are methods that not only consider the initial keying phase where the group establishes the secret key, but also assume subsequent frequent changes to the key as members join and leave the group. An example of an early scalable key management schemes is the One-way Function Tree [23]. The scheme uses *key graphs* [36], logical tree structures of intermediate keys leading up to the group key itself, to efficiently redistribute the new key at the cost of

extra computation and communication during the initial keying phase[1].

Key management functionality is nowadays not restricted to specialized key management schemes – in addition to the initial group key agreement, most new GKA protocols provide auxiliary protocols to handle rekeying on dynamic group events (i.e., members joining or leaving).

2.1 Key Agreement in Groups

Characteristic of the centralized key management schemes mentioned in the previous section is the assumption of pre-existing security associations. If the keys generated by the KDC were transmitted to the members of the group in plaintext, an attacker eavesdropping on the communication channel would be able to obtain the group key. Therefore, members of the group have to share a pairwise key with the KDC in order to obtain those secrets in a confidential manner.

While such an approach is very light on computational requirements placed on the individual group members, it is demanding on the central key distribution server:

Increased storage requirements – a client only has to store the pairwise keys, the group key, and any relevant key encrypting keys from non-leaf nodes on his keypath, a total of $\log_2 n + 1$. In contrast, the server has to store all the pairwise keys and the whole key tree for every group it maintains. Assuming a perfectly balanced binary key tree, this amounts to $2n - 1$ keys per group of size n.

Increased computational burden – the server has to generate all the required keys and encrypt these before sending them to the members of a group.

Increased entropy requirements – as the server is responsible for the generation of the keying material, it must have access to a random or pseudorandom number generator seeded with enough entropy not to produce weak keys.

Unless one can do away with pairwise key establishment prior to the actual key distribution taking place, there will always be a significant overhead in first establishing the group key. This becomes even more costly for

1. The intended application of such schemes is the establishment of secure multicast in very large networks, where the extra overhead associated with setting up the initial key is outweighed by the advantages of fast rekeying.

scenarios where the group forms spontaneously and does not receive any subsequent changes in its structure.

To avoid this overhead, key agreement methods remove the assumption of pairwise secure channels and either use open or authenticated channels. Members of a group that wish to agree upon a secret key do not passively receive a pre-computed value, but rather the *blinded contributions* of the remaining members, obtained by applying a particular one-way function on the contribution, or an intermediate value that contains the contributions of other members instead (typically obtained in the same fashion). They then apply their secret contributions to the value(s) received and send the resulting value(s) to one or more members of the group. The final step then usually consists of all but one users[2] receiving the intermediate value(s) and computing the final group key as a function of the received value(s) and the member's secret contribution.

We now formally establish what group key agreement is, and some of its properties. Probably the first formal definition of group key agreement was given by Burmester and Desmedt[3] in [12]. Ateniese et al. formalized the notion of group key agreement and its properties in [1], based on the definitions of two-party key exchange protocols and their properties in the Handbook of Applied Cryptography. We adopt a slightly modified version of the definitions here.

Definition 1. *A **key agreement protocol** is an n-party protocol that facilitates the establishment of a shared secret computed as a function of the inputs contributed by each of the n parties in such a way that no other party can predetermine the resulting value.*

The term **group key agreement protocol** (GKA) refers to protocols with $n \geq 2$ parties specifically. These are protocols that can handle any number of participants, unlike the two-party protocols usually associated with the term "key exchange".

Definition 2. *A group key agreement protocol is **contributory** if each of the n participating parties contributes fresh information towards the resulting value of the key.*

Definition 3. *Let $\mathcal{P}$ be a group key agreement protocol, $\mathcal{U}$ be the set of protocol participants, and K be the secret key the participants have derived upon executing*

2. There are exceptions to this, as we will see in the Burmester-Desmedt protocol and its modifications.

3. The authors used the term "key exchange" instead of key agreement. The use of any particular term is guided purely by personal preference and both refer to the same thing.

*$\mathcal{P}$. We say that $\mathcal{P}$ provides **implicit key authentication** if each participant $U_i \in \mathcal{U}$ is assured that no party $U_e \notin \mathcal{U}$ can derive the key.*

Definition 4. *Let $\mathcal{P}$ be a group key agreement protocol, $\mathcal{U}$ be the set of protocol participants, and K be the secret key the participants have derived upon executing $\mathcal{P}$. Let $\mathcal{V}_\setminus$ denote the set of peers with whom U_i communicates. We say that $\mathcal{P}$ provides **key confirmation** if upon completion of $\mathcal{P}$ each $U_i \in \mathcal{U}$ is assured that every $V_j \in \mathcal{V}$ is in posession of K.*

Key confirmation does not give any guarrantees about the identity of the parties that are in possession of the key – it only assures the protocol participants that the parties they communicated with are in possession of the key. Key authentication restricts the knowledge of the key to the legitimate members – if a specific party derives the key, it is a legitimate member of the group.

Definition 5. *A group key agreement protocol provides **explicit key authentication** if it provides both implicit key authentication and key confirmation.*

Definition 6. *An **authenticated group key agreement protocol** is a group key agreement protocol that provides implicit key authentication.*

Definition 7. *A contributory group key agreement protocol provides **key integrity** if only legitimate group members who participate in the protocol run are able to contribute information to the resulting secret key.*

Definition 8. *A group key agreement protocol provides **perfect forward secrecy** if compromise of any long-term keys does not result in the compromise of past group keys.*

Definition 9. *A group key agreement protocol is said to be vulnerable to **known-key attack** if compromise of past group keys leads to either (i) compromise of future group keys, or (ii) successful impersonation of one of the members of the group.*

Group key agreement can be built on a number of computationally hard problems, but most frequently, security relies on Decisional Diffie-Hellman assumption in a prime-order, multiplicative cylic group. A few adaptations of existing protocols make use of elliptic curve cryptography, but the security assumptions remain the same. In both cases, protocols often use digital signatures and rely on their security to protect the key agreement against active attackers. Additional assumptions may be made in specific cases; these will be discussed in the next section.

2.2 Attacker Model and Security Assumptions

We assume the standard attacker model as defined by Needham and Schroeder, i.e., an attacker who has complete control over the communication channel and can eavesdrop, inject, modify, reorder or otherwise change the flow of data on the channel. Based on specific capabilities on the part of the attacker, we distinguish between passive attacks and adversaries, and active attacks. *Passive attacks* consist of eavesdropping on the communication channel followed by computation on the observed data with the goal of obtaining the key. *Active attacks* involve manipulation of the channel, with the attacker modifying data sent over that channel or injecting his own; passive attacks may allow certain types of active attacks to be mounted. Our main focus is the security of the group key, and in the case of active adversaries, resilience against outside influence on the key. We specifically exclude the case of malicious insiders and focus on active attackers that never were members of the group.

Security against passive attacks is achieved by basing the key derivation procedure on some computationally hard problem that makes it infeasible to compute the group key using only the information transmitted over the channel. The majority of proposed methods are based on some variant of the Diffie-Hellman problem. To protect against active attacks, protocols have to ensure that messages sent over the channel are not modified, and that messages that were not sent by legitimate group members are rejected. This is usually achieved through the use of digital signature, but protocols based on bilinear pairings that make use of identity-based or certificateless cryptography have become an area of active research in recent years.

There are two main variants of the Diffie-Hellman problem – the Computational Diffie-Hellman and the Decisional Diffie Hellman problem. The original problem, used as a basis for the Diffie-Hellman protocol, is defined in the multiplicative group $\mathbb{Z}_p^*$ for p prime; the generalized versions are defined in finite cyclic groups.

Computational Diffie-Hellman Problem. *Let g be the generator of a finite cyclic group $\mathcal{G}$, and choose $h_1 = g^x, h_2 = g^y \in \mathcal{G}$ at random. Given h_1 and h_2, find g^{xy}.*

The CDH has been shown to be polynomial time-reducible to the discrete logarithm problem [**?**], which is believed to be computationally infeasible. Protocols that rest their security assumptions on CDH never transmit the values x and y. These are considered random secrets the respective parties choose as their inputs, and are never transmitted on the channel. A

passive attacker always "hears" only the blinded secrets, and thus cannot compute the key.

However, the definition of CDH does not rule out the possibility of the attacker correctly guessing some bits of the key. This may allow the attacker to launch a successful known-plaintext attack. It is for this reason that the second assumption is frequently used.

Decisional Diffie-Hellman Problem. *We say the DDH problem is hard relative to $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$ there exists a negligible function* **negl** *such that*

$$|Pr[\mathcal{A}(\mathbb{G}, q, g^x, g^y, g^z) = 1] - Pr[\mathbb{A}(\mathcal{G}, q, g^x, g^y, g^{xy}) = 1]| \leq \mathbf{negl}(n),$$

where in each case the probabilities are taken over the experiment in which $\mathcal{G}(1^n)$ outputs $(\mathbb{G}, q, g)$, and then random $x, y, z \in \mathbb{Z}_q$ are chosen.

Informally, if the value g^{xy} represents a key, the DDH problem states an attacker cannot tell the key from a random one in a reasonable amount of time. The DDH problem gives a strictly stronger guarantee regarding the security of the established key – if the attacker is unable to distinguish a random group element from the secret key, he cannot effectively predict invidual bits of the key. Protocols that rest their security on the DDH assumption thus guarantee semantic security in the presence of a passive adversary.

The DDH problem is not intractable in the same groups as CDH. Boneh [5] gives an example of some groups where DDH is assumed to be intractable, as well as one where CDH is believed to be hard, but DDH is not – the multiplicative group $\mathbb{Z}_p^*$, where p is prime. We list two of these that are relevant for practical implementations that use modular multiplication or elliptic curves.

1. Let p, q be two primes such that $p = 2q + 1$. The DDH is hard in the subgroup Q_p of quadratic residues in $\mathbb{Z}_p^*$.
2. Let E be an elliptic curve of the form $y^2 = x^3 + ax + b$ over the field $\mathbb{F}_p$, where p is a prime. The DDH is hard in E.

2.3 Proofs of Security

Early protocols have frequently employed "heuristic proofs"[4] whereby the protocol is proven secure by considering individual attacks and providing a

4. This is a widely used term in research papers. Possibly the first occurrence can be dated back to [11] where the authors proposed the first formal model for group key agreement protocols.

"rationale" for the fact that the attacker cannot successfully mount such an attack due to the properties of the protocol and/or the underlying security assumptions.

The first move towards formal treatment of proofs of security of authenticated group key agreement protocols was done by Bresson et al. in [11]. The authors define a formal security model of a GKA protocol and specify the actions an adversary, who is modelled to be an outsider, can take to interact with honest protocol participants with the aim of obtaining the key. Protocol execution is then modelled as a game played by the adversary with the participants of the protocol.

Protocol participants are represented by their instances, called *oracles*, that are specific to a given protocol run and independent of each other. Protocol runs themselves can be executed in parallel, so that when the adversary plays the game, he is not only able to have the participants execute multiple protocol runs but also have them interact across different runs as well.

The adversary himself interacts with the participants' oracles through four specific queries – Send, Reveal, Corrupt and Test. These model the adversary's ability, in order of appearance, to hijack the channel, sending arbitrary messages to the participants; reveal an oracle's session (i.e., group) key associated with a particular protocol run; and reveal the oracle's long-term keys, thereby making it possible for the adversary to successfully impersonate an honest participant. The last query models the semantic security of the protocol and is asked only once during the game – the adversary is randomly given either a random bitstring or the group key corresponding to a protocol run of his choice, and has to tell whether the value he was presented is the true key or not.

Ultimately, the goal is to prove that regardless of how much time the adversary has and how many queries to the oracles he makes, he will be able to obtain the group key with only negligible chance[5]. The overall security of the protocol is expressed as a function of hardness of the underlying security assumptions and the resources (number of queries, time) available to the adversary.

When the protocol is authenticated by signatures, additional security assumptions are usually made and modelled. The two most common are the security of the signature scheme against existential forgeries under adaptive chosen-message attacks and the existence of random oracles. *Adaptive*

5. The authors use the term "advantage", which is a function of the adversary's time and number of queries with a range of 0 to 1.

chosen-message attacks scenario refers to the situation where an attacker first learns the signatures on arbitrary messages of his choice, while *existential forgery* corresponds to the attacker's ability to produce a valid signature on some (at least one) message m.

The assumption of existence of a *random oracle* is used in protocols which rely on either hash function or a function that maps n-bit input to $\updownarrow(n)$-bit output randomly, but consistently across queries. Such functions are then replaced by random oracles in proofs – cryptographic entities that behave like random functions and, otherwise from responding to queries x by a random but consistent reply $H(x)$, cannot be examined. Proofs that utilize the random oracle are commonly referred to as proofs under the random oracle model; under the Bresson model, security also becomes parameterized by the number of queries to the random oracle. The idea of using random oracles in proofs was first formulated in [4].

The initial Bresson model was later extended to cover other variants of GKA protocols and attackers. Bresson et al. extended the basic model in subsequent work to cover protocols that facilitate dynamic group changes [8], password-based protocols [10] as well as protocols that did not rely on the existence of random oracles [9].

Chapter 3

Group Key Agreement Protocols

The concept of a protocol that enables members of a group to establish a cryptographic key shared by each member of that group is surprisingly old. Probably the first GKA protocol was proposed by Ingemarsson et al. [16] only six years after the publication of Diffie-Hellman's two-party protocol which essentially defined the concept of key agreement in a two-party setting. The protocol has two specific characteristics, perhaps reminiscent of the beginnings of the field as a whole at that time, that are not present in any other protocol that has been subsequently proposed over the years. First, communication is assumed to be parallel, with the members connected on a logical ring; this poses a challenge as group members thus need to have synchronized logical clock time. Second, the protocol can be parameterized to provide passive security against an adversary that is eavesdropping on a specific number of lines, and the authors establish a measure of resilience of a given network against a passive attacker who wants to decrypt all the messages in that particular network. Modern protocols explicitly assume that the adversary can eavesdrop on all communication channels, and even across multiple protocol runs.

The next early proposal came from Steer et al. [24] six years after Ingemarsson et al. published their protocol. While specifically targetting teleconferencing systems, the logical communication infrastructure is generic enough for the protocol to be used directly in any network in which a group member can send data to the remaining group members, irrespective of any physical topology. The authors explicitly mention this at the end of their paper, noting that the protocol can be used outside teleconferencing systems without any issues. The protocol was later extended to handle dynamic group events by Steiner et al. There, the authors emphasize the specific tree-like way in which the group key is established, and use this (along with the assumption that the group operates on a logical tree) to provide member addition and deletion functionality.

The field became an area of active research in the last decade of the 20th century, and has been receiving continuous attention every since. An im-

portant milestone has been the establishment of lower bounds on communication complexity of GKA protocols that do not handle dynamic group events by Becker and Wille [3]. The authors established the communication complexity of group key agreement protocols in terms of the number of messages, exchanges[1] and rounds for protocols both with and without broadcast. Of particular interest is the lower bound on the number of messages and rounds a GKA protocol has to use – the authors showed that protocols without broadcast must send at least $2n - 2$ messages in at least $\lceil \log_2\ n \rceil$ round, where n is the number of protocol participants. For protocols with broadcast, a minimum of n messages within at least 1 round has to be sent.

Traditionally, GKA protocols were built with modular arithmetics in finite groups where the DDH assumption is assumed to hold; a select few protocols (namely [16], [26], and the authenticated [1] and dynamic variants [25] of the latter of the two) rely on the extended Group DDH assumption that was first introduced in [26]. For data origin authentication, such protocols typically rely on public key cryptography to sign messages exchanged during a protocol run.

The last decade saw a shift in research focus from purely static GKA protocols – where the group is assumed to remain the same – to dynamic GKA protocols that allow for member addition and deletion after the initial group key has been established. Alternate protocol constructions have also been lately explored, with protocols based on bilinear pairings with identity-based or certificateless cryptography gaining increasing research attention in the last decade.

The focus of the rest of this chapter will be on static and dynamic GKA protocols with arithmetic in finite (cyclic) groups that rely on the DDH assumption (or one of its variants) and thus can base their operations in the subgroup of quadratic residues modulo a large prime p. Due to space constraints, dynamic group operations are only briefly explained in the text, as these are not relevant to our scenario and thus will not be implemented. Protocols that have both an authenticated and non-authenticated version are primarily explored in their non-authenticated variant, with the specifics of authentication explained briefly in the text. Alternate protocol constructions are beyond the scope of this thesis.

Throughout the rest of this chapter, we use the following notation in the description of GKA protocols:

---

1. The authors define an **exchange** as a simultaneous exchange of messages between two members in a group, i.e., member A and B each sending a message to each other.

n	number of participants in the protocol
i, j, k	the indices of group members
U_i	i-th group member
G	finite cyclic group of prime order q in which DDH holds
g	generator of G
x_i	U_i's random secret
p, q	large primes (unless otherwise noted)
$\max(\mathrm{g})$	the maximum element in G
K	the group key
σ_i	signature using member U_i's key
ϵ_{e_i}	encryption using member U_i's key

3.1 Protocols on Rings

Ring-based protocols assume that protocol participants are organized in a logical ring and thus know the address of their left and right neighbour. With the exception of the Ingemarsson protocol, all protocols presented in this section assume that members have access to a broadcast channel associated with the whole group.

There are two important results in this category – the GKA protocol proposed by Ingemarsson et al., and the protocol by Burmester and Desmedt [12]. Aside from being probably the first GKA protocol that has been proposed, the former laid the foundation for the work of Steiner et al. [26] as the first protocol that based its security on the Group DDH assumption[2]. The latter is possibly the first constant round GKA protocol, with interesting communication vs. computation complexity trade-off, and later became the subject of some important extensions and improvements [**?**, **?**].

3.1.1 Ingemarsson et al.

A static, non-authenticated GKA protocol was proposed by Ingemarsson et al. in [16]. The protocol has several variants, called *orders* by the authors, that differ in the way the individual contributions are combined in the resultant group key. These combinations are determined by a corresponding *symmetric function*, defined as the sum of all possible products of j variables $R_i, i \in I$ where I is the set of indices of all group members. For $j = |I| = n$

2. Only the order n version of the protocol, however, relies on this assumption. Since that is the only version that is resilient against passive attacks, the remaining ones need not be considered.

the corresponding symmetric function takes the form

$$S^{(j)}(I) = g^{R_0 R_1 \cdot R_{n-1}} \pmod{p};$$

this yields a variant of the protocol (order n) that is both secure against a passive adversary as defined in the previous chapter as well as most computationally efficient out of the whole protocol family. For this reason, we will leave out discussion of the other orders of the protocol.

The order n variant is quite simple – in every time instant, each member receives an intermediate value from his left neighbout, applies his secret to it (by modular exponentiation), and sends this value to his right neighbour. This step is repeated $n-1$ times, with every member computing the key in the n-th time instant by applyng his secret to the value he received. The protocol assumes that the clocks of individual members are synchronized and thus all communication is done in parallel. Protocol 1 presents the Ingemarsson GKA.

Protocol 1 INGEMARSSON ET AL.

COMPUTATIONAL COMPLEXITY: n exponentiations
COMMUNICATION COMPLEXITY: $n-1$ rounds, $n(n-1)$ messages ($n-1$ messages per member), $O(\log_2 p)$ msg. size
SECURITY: Passive (DDH assumption)
PUBLIC DOMAIN PARAMETERS: $G = \mathbb{Z}_p^*, p = 2q+1; p, q$ primes, $g \in \mathbb{Z}_p^*$ of order p
Members are ordered on a ring; indices are taken modulo n so that $U_0 = U_n, U_{i-1} = U_{n-1}$. Every member chooses a random x_i, then executes the following protocol:

ROUND 1: $U_i \rightarrow U_{i+1} : M_1 = g^{x_i} \pmod{p}$
ROUND $j = [2, n-1]$:

1. $U_i \leftarrow U_{i-1} : M_i = g^{x_{i-j+1} x_{i-j} \ldots x_{i-1}} \pmod{p}$
2. $U_i \rightarrow U_{i+1} : M_i^{x_i} = g^{x_{i-j+1} x_{i-j} \ldots x_{i-1} x_i} \pmod{p}$

ROUND n:

1. $U_i \leftarrow U_{i-1} : M_n = g^{x_{i-n+1} x_{i-n+2} \cdots x_{i-1}} \pmod{p}$
2. $K = M_n^{x_i} = g^{\prod_{i=0}^{n-1} x_i} \pmod{p}$

3.1.2 Burmester-Desmedt

Burmester and Desmedt proposed a suite of group key establishment protocols in [12]. The first two protocols are customized for specific network topologies (star and tree), however, they are not contributory as the group leader is tasked with choosing the group key himself. The remaining two target ring topologies without broadcasts and networks with broadcast channels. With the exception of the broadcastless ring variant, all protocols finish in a constant number of rounds. Out of the protocols presented in the paper, the broadcast variant is the most interesting due to its communication to computation complexity trade-off, and has been the subject of later modifications and extensions by other authors in the field. For this reason, we will skip the discussion of the remaining variants as they are either not relevant or interesting for our scenario.

In the broadcast variant, participants are assumed to have access to a broadcast channel associated with the whole group. In the first round, participants first choose a random secret, then broadcast their blinded secrets to the rest of the group. In round 2, every member computes an intermediate value from the blinded secrets from their left and right neighbours, broadcasting the result to the rest of the group. Together with the member's secret, the blinded secrets of the other members and the intermediate values are used to compute the final group key in the third round of the protocol run. Protocol 2 shows the Burmester-Desmedt GKA.

As can be seen from the structure of the group key, the first round of the protocol is slightly "inefficient" as the blinded secret is broadcasted to the rest of the group, but required only by the broadcasting member's left and right neighbours. This has been subsequently modified by Dutta and Barua [14] who use the broadcast variant as an entry point for their proposal.

3.1.3 Katz-Yung

Katz and Yung [18] proposed a *protocol compiler* that transforms any GKA protocol secure against passive attackers into an authenticated GKA protocol secure againt an adversary who controls all channels in the network. The authors used the broadcast version of the Burmester-Desmedt protocol – with operations in a cyclic group of prime order that is assumed to satisfy the DDH assumption – as an example of a protocol that is suitable for compilation, and provide a full formal proof of security in the basic Bresson model that was defined in [11]. The compiled version adds one more round

Protocol 2 BURMESTER-DESMEDT

COMPUTATIONAL COMPLEXITY: 3 exponentiations, $O(n^2)$ multiplications
COMMUNICATION COMPLEXITY: 2 rounds, $2n$ messages (2 broadcasts per member), $O(\log_2 p)$ msg. size
SECURITY: Passive (CDH assumption; later proven to be secure under DDH assumption by Katz and Yung [18])
PUBLIC DOMAIN PARAMETERS: $G = \mathbb{Z}_p; p = \theta(2^{N^c}), c \geq 1; g \in \mathbb{Z}_p$ of order $q = \theta(2^N)$ (not necessarily prime), where N is a security parameter
Members are ordered on a ring; indices are taken modulo n so that $U_0 = U_n$, $U_{i-1} = U_{n-1}$. Every member then executes the following protocol:

ROUND 1:

1. Choose r_i at random and compute $z_i = g^{x_i} \pmod p$
2. $U_i \rightarrow U_{i-1} : z_i$
3. $U_i \rightarrow U_{i+1} : z_i$

ROUND 2:

1. $X_i = (z_{i+1}/z_{i-1})^{x_i} \pmod p$
2. $U_i \rightarrow \text{All} : X_i$

ROUND 3: Compute $K = z_{i-1}^{x_i n} X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i-2} \pmod p =$
$= g^{\sum a_i a_{i+1}} \pmod p$.

of simultaneous broadcasts and requires messages to be signed; this turns the original protocol into an authenticated one.

The general structure of the authentication mechanism in a compiled protocol is as follows: group members first establish their pairs of signing/verification keys (in addition to any other keys established in the base protocol), choose a random nonce and broadcast that nonce along with their identity to the rest of the group. During normal protocol run, members append their signature to every message they send; the data on which the signature is produced, however, also contain the concatenated set of random nonces the members have exchanged in the beginning of the compiled protocol.

As the compiler models security in the Bresson model, its formal description differs somewhat from the compiled version and the above de-

scription of the authentication mechanism given here. All communication is considered in the context of individual instances, messages exchanged during the protocol have a sequence number that designates the number of the message being sent *from that particular member*, and message verification includes checking whether the senderID is in the set of *partner IDs* of the receiver's instance. In practical implementations this would be achieved by checking the identity of the sender on each message received, and using an individual counter for message sequence numbers that would be incremented every time that particular member sends his message. Recipients would then check whether the received message is the next expected one from that particular member.

3.1.4 Dutta-Barua

Building upon the proposals of Katz-Yung and Burmester-Desmedt, Dutta and Barua proposed an authenticated GKA protocol in [**?**]. The base, unauthenticated version is essentially an optimization of the Burmester-Desmedt protocol with operations in a finite cyclic group in which DDH is assumed to hold. The authenticated version is a Katz-Yung compiled version of the base Dutta-Barua protocol; the authors also provide a dynamic version that can handle group changes.

There are two key differences between the Dutta-Barua and Burmester-Desmedt versions of the protocol. First, members do not broadcast their blinded secrets in the first round, sending them to their left and right neighbours instead. In addition to saving bandwidth on networks where broadcasts are technically realized by repeatedly sending the same message to the individual users, this can save important bandwidth and speed the protocol up, as members receive the relevant message faster and do not have to filter out packets they don't need.

Second, the components of the intermediate value in round 2 are computed individually. This results in one extra modular exponentiation but as we will see in the protocol transcript below, also removes the full exponentiation from the third round of the protocol and reduces the overall amount of modular multiplication down to $2n - 2$. The static unauthenticated version of the protocol is shown as Protocol 4.

Dynamic events are handled in a similar fashion. Addition of a group of members is facilitated by essentially executing the protocol anew, but with users U_3 to U_{n-1} not actively participating (i.e., sending messages). Members U_0, U_1 and U_{n-1} assume aliases V_0, V_1, V_2 and run the same protocol with the m incoming members who, for the purposes of the protocol run,

Protocol 3 KATZ-YUNG – COMPILED BURMESTER-DESMEDT

COMPUTATIONAL COMPLEXITY: complexity of Burmester-Desmedt + $(2s + (2n-2) \cdot v)$, where s is the cost associated with creating a signature while v the cost associated with verifying a signature
COMMUNICATION COMPLEXITY: 3 rounds, $3n$ messages (3 broadcasts per member), $O(\log_2 |\max(\mathrm{g})| + \log_2 |\sigma|$ msg. size
SECURITY: Active (DDH assumption, strongly unforgeable signature scheme under adaptive chosen message attack)
PUBLIC DOMAIN PARAMETERS: g of prime order q such that $G = \langle g \rangle$

ROUND 1:

1. Create a pair of signing/verification keys (PK_{U_i}, SK_{U_i}).
2. Choose a random nonce $N_i \in \{0,1\}^k$ and broadcast $U_i|0|N_i$. After receiving broadcasts from all remaining members, each member stores a record of the nonces as $\mathrm{nonces_{U_i}} = U_1|N_1|\cdots|U_n|N_n$.

ROUND 2:

1. Choose $x_i \in \mathbb{Z}_q$ at random and compute $z_i = g^{x_i}$
2. Compute the signatures $\sigma_{i-1} \leftarrow \mathrm{Sign}_{SK_{U_i}}(z_i|U_{i-1}|nonces_{U_i})$ and $\sigma_{i+1} \leftarrow \mathrm{Sign}_{SK_{U_i}}(z_i|U_{i+1}|nonces_{U_i})$
3. $U_i \rightarrow \mathrm{All} : (U_i, z_i, \sigma)$

ROUND 3:

1. Receive (U_j, z_j, σ), $j = \{i-1, i+1\}$. Check that σ is a valid signature on an expected message from member U_j: $\mathrm{Verify}_{PK_{U_j}}(z_j, U_j, \sigma) = 1$.
2. $X_i = (z_{i+1}/z_{i-1})^{x_i}$
3. Compute the signatures $\sigma \leftarrow \mathrm{Sign}_{SK_{U_i}}(X_i|nonces_{U_i})$
4. $U_i \rightarrow \mathrm{All} : (U_i, X_i, \sigma)$

ROUND 4:

1. Receive X_j, $j \neq i$ and verify signatures.
2. Compute $K = z_{i-1}^{x_i n} X_i^{n-1} X_{i+1}^{m-2} \cdots X_{i-2} = g^{\sum a_i a_{i+1}}$.

Protocol 4 DUTTA-BARUA – UNAUTHENTICATED STATIC VERSION

COMPUTATIONAL COMPLEXITY: 3 modular exponentiations + $(2n-2)$ modular multiplications
COMMUNICATION COMPLEXITY: 3 rounds, $3n$ messages (2 messages + 1 broadcast per member), $O(\log_2 |\max(\mathrm{g})|)$ msg. size
SECURITY: Passive (DDH assumption)
PUBLIC DOMAIN PARAMETERS: g of prime order q such that $G = \langle g \rangle$

ROUND 1:

1. Choose x_i at random and compute $X_i = g^{x_i}$
2. $U_i \rightarrow \{U_{i-1}, U_{i+1}\} : X_i$

ROUND 2:

1. $U_i \leftarrow \{U_{i-1}, U_{i+1}\} : X_{i-1}, X_{i+1}$
2. Compute the left key $K_i^L = X_{i-1}^{x_i}$, right key $K_i^R = X_{i+1}^{x_i}$ and the intermediate value $Y_i = K_i^R / K_i^L$.
3. $U_i \rightarrow \text{All} : Y_i$

ROUND 3:

1. $U_i \leftarrow \text{All} : Y_j, j \in \{0, \ldots, n-1\} - \{i\}$
2. Iteratively compute $\overline{K}_{i+1}^R, \overline{K}_{i+2}^R, \ldots, \overline{K}_{i+(n-1)}^R$ using the following formula:

$$\begin{aligned} \overline{K}_{i+1}^R &= Y_{i+1} K_i^R, \\ \overline{K}_{i+j}^R &= Y_{i+j} \overline{K}_{i+(j-1)}^R. \end{aligned}$$

3. Check whether $\overline{K}_{i+(n-1)}^R = K_i^L (= K_{i+(n-1)}^R)$. If no, abort.
4. Compute $K = \overline{K}_0^R \overline{K}_1^R \ldots \overline{K}_{n-1}^R = g^{x_0 x_1 + x_1 x_2 + \ldots + x_{n-1} x_0}$.

are labeled V_4 to V_{m+3}. Instead of choosing a random secret, however, U_0 and U_2 reuse their old ones while U_1 uses the seed $s = H(K)$ (where H is a hash function) as its secret. In the first round, $U_0 = V_0$ and $U_{n-1} = V_2$ also send their blinded secret to the remaining members $U_3, \ldots, U_{n-1}$; these members additionally receive all broadcasts from round 2 of the procotol. Given this information, they can compute the new key.

Member deletion is an even simpler operation – the left and right neighbours of the leaving members choose a new secret and run the base protocol with the remaining users, who reuse their old secrets.

3.1.5 Generic n-party Diffie-Hellman – GDH.1-3 protocols

Another fundamental result in the field was the publication of the GDH protocol suite by Steiner et al. in [26]. The three proposed protocols, GDH.1-GDH.3, rely on a new security assumption that the authors have introduced in the paper– the Group DDH (G-DDH) assumption[3], which is in essence a variant of the DDH assumption for keys that have the form $g^{x_1 x_2 \ldots x_n}$.

More formally, given the set $\{g^{\prod x_i}\}$ of elements with a subset of indices $i \in \{1, \ldots, n\}$, the goal is to distinguish $g^{x_1 x_2 \ldots x_n}$ from a random value. The authors prove that if the DDH holds in a group, the G-DDH holds as well; choosing a group in which the DDH holds is therefore enough to guarantee the same level of security against passive attacks.

The structure of the three proposed protocols closely mirrors this security assumption – throughout the protocol run, group members are building and exchanging a set of these elements, obtaining the resulting group key in the last round in which they each use their secret contribution to compute the key. Each of the protocols is optimized with respect to some characteristics as per the lower bounds on protocol complexity as established by Becker and Wille. For communication purposes, all three protocols assume group members are ordered on a ring, but it is perfectly enough if members lie in a path, with either the last member (in the case of GDH.2) or both the last and the penultimate (in the case of GDH.3) being able to broadcast messages to the rest of the group.

The first protocol, GDH.1, does not use broadcasts, and while running in a number of rounds linear to the number of group members – $2(n-1)$ – it meets the lower bound on the total number of messages sent in a GKA protocol without broadcasts. The protocol can be broken down to two phases,

3. This is a term that has, to the best of our knowledge, been first used several years after the publication of the result by Bresson et al. in [11]. Steiner and his colleagues did not use this term in their paper; in fact, they did not give a specific name to the assumption at all

an *upflow* and a *downflow*. In the upflow phase, every round a member receives a set of intermediate values from his right neighbour (the first member simply uses the generator of the cyclic group as the "received" value), contributes his secret to a specific value from that set, and sends the old set together with the new value to his right neighbour. Once such a set reaches the last member, he knows enough to compute the key in a similar fashion. The downflow phase is then initiated with the same round structure but going from the last to the first member, applying that member's secret to *all* intermediate values instead. In doing so, the member obtains the group key from one of the received intermediate values; the group key is then removed from that set of modified values before the set is sent to the member's left neighbour. Protocol 5 shows the GDH.1 protocol.

Protocol 5 GDH.1

COMPUTATIONAL COMPLEXITY: $(i+1)$ modular exponentiations for $U_i, i \leq n$, n for U_n
COMMUNICATION COMPLEXITY: $2(n-1)$ rounds, $2(n-1)$ messages (2 per member), $O(i \cdot \log_2 |\max(g)|)$ msg. size
SECURITY: Passive (DDH/GDH assumption)
PUBLIC DOMAIN PARAMETERS: g of prime order q such that $G = \langle g \rangle$
Every member U_i chooses a random x_i and executes the following:

ROUND $i; i \in [1, n-1]$:

1. $U_i \rightarrow U_{i+1} : \{X_j | j \in [1, i]\}$ where $X_j = g^{x_1 x_2 \ldots x_j}$

ROUND n:

1. Compute $K = (X_{n-1})^{x_n} = g^{x_1 x_2 \ldots x_n}$.

2. $U_n \rightarrow U_{n-1} : \{Y_j | j \in [1, n-1]\}$ where $Y_j = (X_j)^{x_n}$.

ROUND $n+i; i \in [1, n-1]$:

1. Compute $K = (Y_{n-i})^{x_{n-i}} = (g^{x_1 x_2 \ldots x_{n-i-1} x_{n-i+1} \ldots x_n})^{x_{n-i}} = g^{x_1 x_2 \ldots x_n}$.

2. $U_{n-i} \rightarrow U_{n-i-1} : \{Y_j | j \in [1, n-i-1]\}$

The second protocol, GDH.2, makes use of one broadcast, optimizing both the total number of rounds and messages sent to n. Similarly to GDH.1, we can recognize two phases in the protocol – upflow and broadcast. In the upflow phase, members build up a set of intermediate values; this time,

however, a member applies his random secret to all the values in the set he received. Just like in GDH.1, when the set of intermediate values reaches the last member, he computes the group key immediately, but instead of sending the set of modified intermediate values to his left neighbour, he simply broadcasts it to the rest of the group. Every member then finds the corresponding intermediate value and obtains the group key by applying his secret to it.

Protocol 6 GDH.2

COMPUTATIONAL COMPLEXITY: $(i+1)$ modular exponentiations for $U_i, i \leq n$, n for U_n
COMMUNICATION COMPLEXITY: n rounds, n messages (1 per member), $O(i \cdot \log_2 |\max(\mathrm{g})|)$ msg. size
SECURITY: Passive (DDH/GDH assumption)
PUBLIC DOMAIN PARAMETERS: g of prime order q such that $G = \langle g \rangle$
Every member U_i chooses a random x_i and executes the following:

ROUND $i; i \in [1, n-1]$:

1. $U_i \rightarrow U_{i+1} : \{g^{\prod\{x_k | k \in [1,i] \wedge k \neq j\}} | j \in [1, i]\}, g^{x_1 x_2 \ldots x_i}$

ROUND n:

1. $U_n \rightarrow \text{All} : \{g^{\prod\{x_k | k \in [1,n] \wedge k \neq i\}} | i \in [1, n]\}$

Finally, the GDH.3 protocol optimizes the amount of computation a member has to do throughout the protocol. Utilizing two broadcasts, members compute the key in a linear number of rounds $(n+1)$ but in a constant number of modular exponentiations – except for the last member, who has to perform exactly n exponentiations. The protocol starts in a fashion similar to GDH.1 and GDH.2 – members exchange intermediate values starting from the first one up until the penultimate one.

This time, however, only a single value is being passed between two members in a round; this value contains the secrets of all the members preceding the receiving member in the given order. Once the intermediate value reaches the penultimate user, he applies his random secret and broadcasts the resulting value to the rest of the group, including the last member. The last member now knows enough to compute the group key, while the other members, including the penultimate one, strip their secret from the intermediate value and send it to the last member. The last mem-

ber then applies his secret to the received values and broadcasts them all in a batch. The remaining users find their corresponding intermediate value in the set of received values and re-apply their secret to obtain the group key.

Protocol 7 GDH.3

COMPUTATIONAL COMPLEXITY: 3 modular exponentiations for $U_i, i \leq n$; n for U_n
COMMUNICATION COMPLEXITY: $n+1$ rounds, $2n-1$ messages (2 per member $U_i, i \neq \{n-1, n\}$; 1 message + 1 broadcast for U_{n-1}, 1 broadcast for U_n), $O(\log_2 |\max(\mathrm{g})|)$ msg. size
SECURITY: Passive (DDH/GDH assumption)
PUBLIC DOMAIN PARAMETERS: g of prime order q such that $G = \langle g \rangle$
Every member U_i chooses a random x_i and executes the following:

ROUND $i; i \in [1, n-1]$:

1. $U_i \rightarrow U_{i+1} : g^{\prod\{x_k | k \in [1, i-1]\}}$

ROUND $n-1$:

1. $U_{n-1} \rightarrow \text{All} : g^{\prod\{x_k | k \in [1, n-1]\}}$

ROUND n:

1. $U_i \rightarrow U_n : g^{\prod\{x_k | k \in [1, n-1] \wedge k \neq i\}}$

ROUND $n+1$:

1. $U_n \rightarrow \text{All} : \{g^{\prod\{x_k | k \in [1, n] \wedge k \neq i\}} | i \in [1, n-1]\}$

In their paper, the authors also provide a sketch of a possible extension to the base protocols to handle basic dynamic group events – member addition and deletion. These ideas were then developed further in their subsequent publications.

The CLIQUES protocol suite [25] developed a set of auxiliary protocols to handle member addition and deletion, group fusion, subgroup exclusion and mass join[4], using the GDH.2 protocol to establish the initial group key.

4. The authors differentiate between the scenario where a set of members who already form a group with their own group key wishes to join another group of members with its own key (group fusion), and a set of users with no particular association between them wishing

3.1.6 Authenticated Extensions of GDH

In a later publication [1], Ateniese et al. have proposed authenicated versions of the GDH.2 protocol. The first variant uses pairwise keys K_{in} shared between members $U_i, i \leq n-1$ and U_n to provide implicit key authentication. The authenticated version is the same as the unauthenticated GDH.2 except for the last round, in which the pairwise keys are added to the intermediate values member U_n broadcasts to the rest of the group:

$$U_n \rightarrow \text{All} : \{g^{\frac{x_1 x_2 \ldots x_n}{x_i} K_{in}} | i \in [1, n-1]\}.$$

The receiving members then compute the group key by raising their corresponding intermediate value to their random secret and the pairwise key's inverse: $K = g^{(\frac{x_1 x_2 \ldots x_n}{x_i} \cdot K_{in}) \cdot K_{in}^{-1} \cdot x_i} = g^{x_1 x_2 \ldots x_n}$.

As can be seen, the key is only used as a mechanism to exclude an outside party from computing the group key. However, it is not possible to exclude said party from influencing the resulting group key during the first $n-1$ rounds. The second authenticated variant of GDH.2 proposed in the paper tries to amend this by using pairwise keys shared among *all* members of the group in every round.

Bresson et al proposed a number of provably secure authenticated variants of the GDH.2 protocol in [11, 8, 10, 9]. In their first paper, the authors defined the security model we described in Chapter 2 and presented an authenticated version of the static GDH.2 protocol which they then proved to be secure under the random oracle model.

The authenticated protocol is essentially the same as GDH.2, however, messages also contain a list of all group members' identities and are now required to be signed. As a result, the protocol provides entity authentication as well. The authors also describe an extension that provides key confirmation functionality, and prove the extended protocol to be securel. Concluding the list of modifications is the use of a hash function to derive the session key as $sk = \mathcal{H}(U_1, U_2, \ldots, U_n, Fl_n, g^{x_1 x_2 \ldots x_n})$, where Fl_4 is the message broadcast by member U_n in the last round of the protocol to the rest of the group.

The basic security model was then subsequently extended to handle dynamic events in [8, 9]. In [8], the authors extended the authenticated version of the GDH.2 protocol to handle dynamic events, with a form proof of security under the random oracle model. In [9], the authors further modified the authenticated variant of the GDH.2 protocol, proving it secure under the

---

to join an existing group with its own key (mass join).

standard model. Among some of the extensions is the explicit use of smart cards to hold ElGamal-like signature keys, secure coprocessors to generate the random secrets, and universal hash functions to derive the session keys.

Finally, the authors presented a password-based variant of the GDH.2 protocol that is secure against dictionary attacks in [10]. The basic structure of the protocol remains unchanged, but password-based encryption is used instead of digital signatures on messages. Members also use two random secrets $x_i, \nu_i \in [1, q-1]$ instead of one so that now the exchanged messages have the form

$$\varepsilon_{pw}(\{g_i^{\prod\{x_k|k\in[1,i]\wedge k\neq j\}}|j \in [1,i]\}, g_i^{x_1x_2\ldots x_i}),$$

where $g_i = g^{\nu_1\nu_2\ldots\nu_i}$ and ε_{pw} denotes the password-based encryption. The session key is then computed as $sk = \mathcal{H}(U_1, U_2, \ldots, U_n, Fl_n, K)$, where $K = g_n^{x_1x_2\ldots x_n}$.

3.2 Tree-based protocols

Tree-based protocols assume the protocol participants are organized on a logical tree. Depending on the protocol, the tree can be either balanced or unbalanced. Tree-based protocols explicitly focus on making rekeying during dynamic group events efficient. We only cover two protocols in this section – the protocol by Perrig [22] and STR – as these have a dedicated GKA protocol as well.

3.2.1 (N)AGKA

Perrig proposed several variants of a tree-based protocol in [22]; we will present the unauthenticated version here. The unauthenticated protocol, called NAGKA, assumes that the size of the group is 2^d and thus members can build a balanced binary tree. Group members assume positions as leafs in the tree and subsequently build up the rest of the tree from the bottom up. Intermediate nodes then hold keys shared by those members who are positioned as leafs in a subtree rooted at the intermediate node. The group key is contained at the root of the whole tree. Figure 3.1 shows an example of a balanced key tree for a group of 4 members.

To compute the group key, members iteratively construct keys for the intermediate nodes from the bottom up. To compute the key $K_{\langle l,v\rangle}$ for an intermediate node v at level l, two members are chosen from its left and right subtree randomly – one from each – and broadcast the keys $K_{\langle l+1,v_1\rangle}, K_{\langle l+1,v_2\rangle}$ associated with the node v's left and right children v^L, v^R to the rest of the

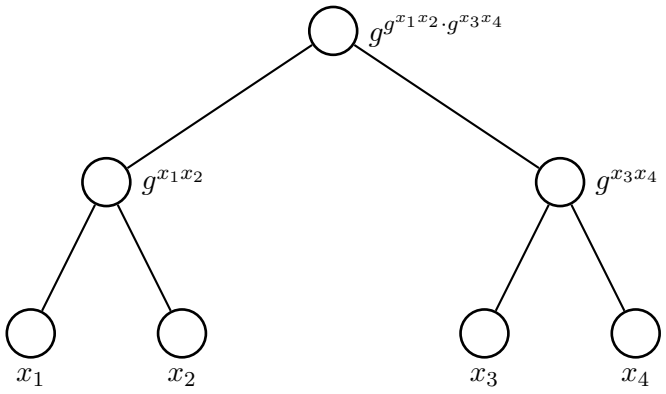

Figure 3.1: An example balanced logical key tree for 4 members after a successful NAGKA run.

group (blinded member secrets g^{x_i} are used as node keys on the leaf level). Using these values, the remaining members of the subtree rooted at node v can now compute the key as $g^{K_{\langle l+1,v_1\rangle} \cdot K_{\langle l+1,v_2\rangle}} \pmod p$. Protocol 8 shows the protocol run.

The authenticated version uses digital signatures to sign the broadcasts, which now also contain a recent timestamp and the identity of the sender. Dynamic events are handled through changes to the tree, adding new nodes or removing old ones in such a way as to least disbalance the tree. We refer the interested reader to [22] for further details.

3.2.2 STR

Steer et al proposed a GKA protocol for secure teleconferencing in [24]. The protocol was generic enough to be used in any setting where messages could be broadcast to the rest of the group, and was later extended by Kim et al. [19] who described the underlying "skinny tree" structure that is effectively built up during the execution of the protocol. This was then used to handle rekeying operations during member addition or deletion. We will skip the discussion of the skinny tree and the dynamic operations here and refer the interested reader to the authors' publication.

We assume that there exists a linear ordering on the set of members. The protocol starts with each member choosing a random secret, blinding it and broadcasting the blinded value. After this step, the first and the second

Protocol 8 NAGKA

COMPUTATIONAL COMPLEXITY: d modular exponentiations, where $d = \log_2 n$; n is the number of group members
COMMUNICATION COMPLEXITY: d rounds, $2n - 1$ messages (at most d broadcasts per member, 1 minimum), $O(\log_2 |\max(\mathrm{g})|)$ msg. size
SECURITY: Passive (CDH)
PUBLIC DOMAIN PARAMETERS: g of prime order q such that $G = \langle g \rangle$
Node levels are labeled from $d - 1$ (leaf level) to 0 (root).

ROUND 1:

1. Every U_i chooses $x_i \in \mathbb{Z}_q^*$ at random.
2. $U_i \rightarrow$ All : g^{x_i}
3. U_i computes the key for its parent intermediate node v' using the blinded secret x_j of its sibling: $K^{x_j}_{\langle d-2,v' \rangle}$.

ROUND $i; i \in [2, d]$:

1. For every node v on level $l = d - i$, choose U_i, U_j randomly from v's left and right subtrees, respectively.
2. $U_i \rightarrow$ All : $g^{K_{\langle l+1,v^L \rangle}}$
3. $U_j \rightarrow$ All : $g^{K_{\langle l+1,v^R \rangle}}$
4. Members of the subtree rooted at v compute $K_{\langle l,v \rangle} = g^{K_{\langle l+1,v^L \rangle} \cdot K_{\langle l+1,v^R \rangle}}$. For $i = d$, the group key is $K = K_{\langle 0,v \rangle}$.

member is able to compute the group key as

$$K = g^{x_n g^{x_{n-1} g^{\cdots g^{x_1 x_2}}}}$$

while the remaining members each need an intermediate value from one or more members that precede them in the ordering to compute the key. Protocol 9 formally describes the protocol run.

Steer et al. also describe an additional step during which the members exchange their certificates, sign a hash of the set of values $y_i, i \in [1, n]$ which they received in the first round of the protocol run, and broadcast it. This is presumably done in order to provide key confirmation, but since users U_3 to U_n are dependent on intermediate values z_i as well, those would have to

be confirmed. Since this is not so, we consider it safe to skip this additional step and thus decrease the total communication and computation cost.

Protocol 9 STR

COMPUTATIONAL COMPLEXITY: n modular exponentiations, where n is the number of group members
COMMUNICATION COMPLEXITY: $n-1$ rounds, $2n-2$ messages (1 broadcast for U_1, U_n; 2 broadcasts for $U_i, i \in [2, n-1]$), $O(\log_2 p)$ msg. size
SECURITY: Passive (DITGDH/DDH assumption)
PUBLIC DOMAIN PARAMETERS: g of prime order q such that $G = \langle g \rangle$

ROUND 1:

1. Every U_i chooses $x_i \in \mathbb{Z}_q^*$ at random.
2. $U_i \rightarrow$ All : $y_j = g^{x_i}$
3. U_1 computes the group key iteratively for $j = 2$ to n as

$$\begin{aligned} K_1 &= x_1 \\ K_j &= y_j^{K_{j-1}}, \end{aligned}$$

where $K = K_n$.

ROUND $i; i \in [2, n-1]$:

1. Member U_i receives z_{i-1} from U_{i-1} (for U_2, $z_1 = y_1 = g^{x_1}$) and computes $K_i = z_i^{x_i}$.
2. $U_i \rightarrow$ All : $z_i = g^{K_i}$
3. U_i computes the group key iteratively for $j = i+1$ to n as

$$K_j = y_j^{K_i},$$

Member U_n can then compute the group key as $K = z_{n-1}^{x_n}$ after he receives z_{n-1} from U_{n-1}.

Kim et al. have proposed a slightly modified version of the base GKA protocol – instead of members $U_i, i \in [2, n-1]$ computing and broadcasting the values z_i, member U_1 computes those values himself and broadcasts them to the rest of the group. This allows the remaining members to compute the group key at once without having to wait for the values to arrive

independently, albeit at the cost of increased computation on the part of U_1. The authors have also formalized the security assumption of the base protocol, which the authors call the Decisional Tree Imbalanced Group Diffie-Hellman (DITGDH) assumption, and proved that DDH implies DITGDH.

Manulis later presented an elliptic curve version of the dynamic STR protocol in [20, 21]. A unique feature of the protocol was the ability to spread computational load during dynamic group events based on the performance level of individual members, with more computationally powerful members bearing a higher computational cost.

3.3 Star-based Protocols

Star-based protocols assume that members form a logical star topology, with the member in the center designated as a leader that is tasked with extra computational and communication duties. The leader typically collects the individual members' contributions, applies his own contributions to the collected values, and sends the resulting set back to the group. Such a communication model allows the protocols to finish their run in a constant number of rounds. Leadership in star-based protocols is not static, however, and the same group can have a different leader every time its members run the GKA protocol.

3.3.1 Boyd-Nieto

A very compact, static three-round GKA protocol was proposed by Boyd and Nieto in [6]. Group members (including the leader) each have their own public/private signing and encryption keys. The leader U_1 starts the protocol run by picking a random nonce N_1 of k bits, where k is a security parameter, and broadcasting the set of member identities, along with a signature on the set of identities and a set of encryptions $\varepsilon_{e_i}(N)$ of N under the public encryption keys e_i of regular group members.

The leader then proceeds to broadcast the set of encryptions of N to the group members. Finally, group members choose a random k-bit nonce N_i and broadcast the nonce, along with their identity, to the rest of the group. The group key is then computed as $K = \mathcal{H}(U_1||U_2||\ldots||U_n)$, where $\mathcal{H}$ is a hash function and $||$ denotes concatenation.

Assuming a scenario where all the participants are honest, all three stages of the protocol can be fully parallelized, with the leader sending the two messages together and the group members broadcasting their chosen nonces at the same time. Protocol 10 summarizes the protocol run.

Protocol 10 BOYD-NIETO

COMPUTATIONAL COMPLEXITY: $n-1$ public key encryptions + 1 signature generation (leader) + 1 hash, 1 signature verification + 1 hash (regular member)
COMMUNICATION COMPLEXITY: 1 round; $O(n \log_2 l)$ msg. size (leader) where l is the size of the ciphertext, $O(k)$ for regular member
SECURITY: Active (Secure signature and encryption scheme, hash function)
PUBLIC DOMAIN PARAMETERS: k

Members pick a random k-bit nonce and execute the following:

1. $U_1 \rightarrow \text{All} : \mathcal{U} = \{U_1, U_2, \ldots, U_n\}, Sig_1(\varepsilon_{e_2}(N_1), \ldots, \varepsilon_{e_n}(N_1))$
2. $U_1 \rightarrow \text{All} : \mathcal{U}, \varepsilon_{e_i}(N_1), i \in [2, n]$
3. $U_i \rightarrow \text{All} : U_i, N_i$

Group key is then computed as $K = \mathcal{H}(N_1||N_2||\ldots||N_n)$, where $\mathcal{H}$ is a hash function.

The authors modeled the security of the protocol in the extended Bellare-Rogaway model, which is similar to the Bresson model, and proved the protocol secure under the random oracle model. Unfortunately, the protocol does not provide perfect forward secrecy.[5]

3.3.2 Bresson et al.

Bresson et al. proposed a dynamic authenticated GKA protocol for computationally imbalanced groups in [7]. The target scenario for their protocol was a group of low-power mobile devices (members U_1 to U_n) together with a more powerful base station (leader S). Computational load is shifted to the base station so that the mobile devices only perform a few operations. While the protocol is optimized for cheap rekeying operations during dynamic group events, it is generic enough to be used in any scenario where a computationally more powerful member is part of the group.

The protocol assumes that the regular members have a public/private signing keypair, denoted as PK_i, SK_i respectively. The base station has a special "public key" of the form $y = g^x$ where x serves as the station's contribution towards the group key – in the specification of the protocol,

5. The authors were aware of the fact and explicitly stated this in their proposal.

this value is known in advance.

Finally, there are three hash functions: $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l$, where l is the binary length of the prime q that is the order of G in which arithmetic is based; $\mathcal{H}_0 : \{0,1\}^* \rightarrow \{0,1\}^{l_0}$, where l_0 need not equal l; and $\mathcal{H}_1 : \{0,1\}^{l_1} \rightarrow \{0,1\}^{l_0}$, where l_1 is the maximum binary length of the counter c that is used in the protocol to prevent replay attacks.

Members $U_i, i \in [1,n]$ can initiate the protocol run by choosing a random $x_i \in \mathbb{Z}_z^*$, computing $y_i = g^{x_i}, \alpha_i = y^{x_i} = g^{xx_i}$ and sending the signed y_i to the base station.

The values α_i effectively represent a standard two-party DH key established between members U_i and the base station. They serve two purposes – they are used by the base station for the construction of the group key while also serving as an "encryption value" together with the base station's counter.

The base station computes the values $\alpha_i = y_i^x$ from the received values and initializes an l_1-bit counter $c = 0$ (the counter is increased during any subsequent rekeying after a dynamic group event). The station then computes the *shared secret data* as $K = \mathcal{H}_0(c||\alpha_1|| \ldots ||\alpha_n$. The shared secret data are then used by the station to compute $K_i = K \oplus \mathcal{H}_1(c||\alpha_i)$, which are then each sent together with the current value of the counter c to the corresponding member.

Members can extract the shared secret data easily by computing $K = K_1 \oplus \mathcal{H}_1(c||\alpha_i)$. The group key is then computed as $sk = \mathcal{H}(K||\mathcal{U}||S)$, where $\mathcal{U}$ is the set of identites of members $U_i, i \in [1,n]$ and S is the identity of the base station. Protocol 11 presents the GKA protocol.

Dynamic events are handled in a straightforward fashion - when members leave the current group, the base station increases the counter to c', and performs the actions in round 2 for the remaining users in the group. Rekeying after member addition is identical, with the joining members also performing round 1 of the protocol.

Security is modeled in a slightly modified version of the dynamic Bresson model, with a proof under the random oracle model. Since the station's secret key x is treated as a long-term secret, the protocol does not provide perfect forward secrecy as the compromise of x leads to the compromise of the whole session key. This can be avoided if the base station picks x randomly during each protocol run and instead treats y as a blinded secret which is sent digitally signed to the other group members. The protocol is, however, resilient against known key attacks. The authors also present (and prove secure) an extension of the protocol which adds key confirmation to the protocol, at the expense of extra two rounds of communication.

Protocol 11 BRESSON ET AL.

COMPUTATIONAL COMPLEXITY: 2 modular exponentiations + 1 signature generation (member) + 1 hash, n modular exponentiations + n signature verifications + $n + 1$ hashes
COMMUNICATION COMPLEXITY: 2 rounds; $O(\log_2 |\max(\mathrm{g})|)$ msg. size (member), $n \cdot (l_0 + l_1)$ msg. size (leader)
SECURITY: Active (Secure signature scheme, hash functions, CDH)
PUBLIC DOMAIN PARAMETERS: g of prime order q such that $G = \langle g \rangle$; $y = g^x$ the public blinded value of the base station

ROUND 1:

1. Each U_i picks $x_i \in \mathbb{Z}_q^*$ at random, computes $y_i = g^{x_i}, \alpha_i = y^{x_i}$.

2. $U_i \rightarrow S : y_i, \sigma_i(y_i)$, where σ_i denotes the signature using PK_i.

ROUND 2:

1. S verifies the signatures on incoming messages, sets $c = 0 \in \{0,1\}^{l_1}$ and for $i \in [1, n]$ computes the following:

$$\begin{aligned} \alpha_i &= y_i^x \\ K &= \mathcal{H}_0(c||\alpha_1|| \ldots ||\alpha_n) \end{aligned}$$

2. $S \rightarrow U_i : c, K_i = K \oplus \mathcal{H}_1(c||\alpha_i)$

Group key is then computed as $sk = \mathcal{H}(K||\mathcal{U}||S)$, where $\mathcal{U}$ is the set of identities of members U_1 to U_n.

3.3.3 Augot et al.

Augot et al. proposed a dynamic authenticated GKA protocol in [2]. The protocol starts with the leader, chosen randomly from the group members, initializing the protocol run by generating a random k-bit nonce, with k a security parameter, and sending the nonce along with an initialization string[6] signed to the rest of the group.

Individual members then verify the leader's signature on the received message, generate their own nonce and a random secret $x_i \in [1, q-1]$ where q is the prime order of the group G in which arithmetic is based,

6. This is just a string that indicates to the rest of the group that the leader is wishing to start the protocol; its form will depend on the particular implementation.

and compute g^{x_i}. They then respond to the init message by concatenating the leader's message with their nonce and their blinded secret, and sending the resultant message, signed with their key, back to the leader.

The leader collects the messages from the members, verifies their signatures, chooses his own random secret $x_l \in [1, q]$ and raises the received blinded secrets to it by computing $(g^{x_i})^{x_l}$ for each i. The leader then sends back to the members a signed set of the intermediate values together with the other member's individual contributions.

Upon receipt, members verify the leader's signature on the received data and if correct, extract the leader's blinded secret from their corresponding intermediate value by computing $(g^{x_i x_l})^{x_i^{-1}}$. Using the set of intermediate values along with the leader's blinded secret, they can now compute the group key by multiplying the values with the leader's blinded secret, obtaining $K = g^{x_l + \sum_i x_i x_l}$. Protocol 12 shows the complete GKA protocol.

Rekeying during member join and leave events follows the same basic structure as the GKA protocol, but with a different round 1 initiatior – in the join scenario, incoming members initiate rekeying by broadcasting their chosen nonce and blinded secret to the group, while in the leave scenario, the leaving members broadcast their nonce.

The rest of the rekeying procedure is then straightforward – in a member leave event, the leader will choose a new random secret and construct a message analogous to msg_l^2 from the GKA protocol for the remaining members. In a member join event, the leader will instead resend the round 3 message msg_l^2 to a new randomly chosen leader in the now extended group, who will then complete the protocol run by executing round 3 of the GKA protocol on the received data.

Security is modeled in the dynamic Bresson model, with a proof in the standard model.

3.4 Comparison

There are three protocol characteristics with regards to which we can compare the GKA protocols presented in this chapter – computational complexity, communication complexity, and the number of rounds. As in the previous sections, we specifically treat broadcast as a single message.

In terms of computational complexity, we consider digital signatures generation and verification to be the most expensive operation, followed by public-key encryption, exponentiation, multiplication, and computation of hash values. As a result, authenticated variants of protocols are more ex-

pensive than their unauthenticated variants. Among unauthenticated GKA protocols, the Dutta-Barua ring-based protocol is the most computationally efficient, with only three exponentiations and $2n - 2$ multiplications per member required during the protocol run to compute the key.

GDH.3, as well as the star-based protocols by Boyd-Nieto, Bresson et al. and Augot et al. provide an interesting trade-off, shifting the burden of heavier computation to a single member, allowing the rest of the group to compute the key in fewer operations. Particularly in the case of Boyd-Nieto, members are only required to verify a single signature, decrypt one value and compute a hash on the received nonces.

On the other hand, the protocol by Ingemarsson et al. is the most computationally ineffective of the protocols presented in this chapter, with exactly n modular exponentiations required for every member to compute the key.

A number of protocols covered in this chapter need only a constant number of rounds to finish their run. Of these, the Boyd-Nieto protocol allows the members to establish the group key in just a single round, if the leader sends both his messages at once at the same time as the individual members broadcast their chosen nonces. The Burmester-Desmedt, Dutta-Barua and the star-based Bresson et al. allow the group to establish a key in two rounds. The protocol by Augot et al. establishes the key in three rounds, although the first round mostly serves as a mechanism to initiate the protocol run.

In terms of the total number of messages, the GDH.2 and GDH.3 protocols offer the best results, with both computing the resulting group key in just n exchanged messages. The other ring-based protocols do not fare so well, needing at least $2n - 2$ messages (in the case of GDH.1) to establish the key. By shifting part of the computational burden and not communicating between regular group members directly, star-based protocols are able to make an interesting trade-off, with the Boyd-Nieto protocol potentially finishing in just n messages, closely followed by Augot et al. with only $n+1$ overall messages. The star-based Boyd-Nieto requires $2n-2$ messages, making it comparable to ring-based protocols like $Burmester - Desmedt$ or $GDH.1$.

Protocol 12 AUGOT ET AL.

COMPUTATIONAL COMPLEXITY: 2 modular exponentiations + 1 signature generation and verification + n multiplications (member), n modular exponentiations + $n-1$ signature verifications + 1 signature generation n multiplications (leader)
COMMUNICATION COMPLEXITY: 3 rounds; $O(\log_2 |\max(\mathrm{g})|)$ msg. size (member), $n \log_2 |\max(\mathrm{g})|$ msg. size (leader)
SECURITY: Active (Secure signature scheme, DDH assumption)
PUBLIC DOMAIN PARAMETERS: g of prime order q such that $G = \langle g \rangle$

ROUND 1:

1. Choose a leader U_l from the group members randomly. U_l then generates a random nonce $N_l \in \{0,1\}^k$.

2. $U_l \rightarrow \text{All} : msg_l^1 = \{INIT, U_l, N_l\}, \sigma_l^1$

ROUND 2:

1. Each U_i verifies the signature on msg_l^1; if correct, chooses a random nonce $N_i \in \{0,1\}^k$ and random secret $x_i \in [1, q-1]$.

2. $U_i \rightarrow U_l : msg_i = \{IREPLY, U_l, N_l, U_i, N_i, g^{x_i}\}, \sigma_i$

ROUND 3:

1. U_l verifies the signature on each msg_i; if correct, chooses a random secret $x_l \in [1, q-1]$

2. $U_l \rightarrow \text{All} : msg_l^2 = \{IGROUP, U_l, N_l, \{U_i, N_i, g^{x_i}, g^{x_i x_l}\}_{i \in \mathcal{U}}\}, \sigma_l^2$

Upon receipt, members verify the signature on msg_l^2; if correct, they compute the key as $K = g^{x_l + \sigma_{i \in \mathcal{U} \setminus \{l\}} x_i x_l}$. The leader computes the key straight after round 3.

Chapter 4

Implementation

The choice of a specific protocol for implementation is governed by two factors – the computational capabilities of the target systems and the communication channels, with their characteristics, that are available on them. While it may be desirable to have the lowest possible amount of computation and communication in a GKA protocol, it is better to make a trade-off between the number of computation and communication performed in a case where one or the other is more expensive.

Mobile phones today fall into one of two categories – smartphones and feature phones. *Feature phones* are mobile phones that offer basic phone functionality, such as the ability to initiate and receive calls and send text or multimedia messages, together with a set of specific features, such as the ability to play FM radio or music, or take photographs with a built-in camera. Such devices are typically intended as a "multiple purpose device" instead of a general purpose computing platform, with less powerful hardware and simpler software platforms at their core.

Some feature phone platforms, however, allow for the installation of small third-party software on the device, extending the core functionality of the phone somewhat. An example of such a platform is the widespread Nokia S40 which allows Java ME applications to be installed and run on an S40 device.

Smartphones offer a more sophisticated environment, with an operating system that allows the phone to be used more as a portable general purpose computer than a simple single-functionality device. Smartphones typically offer multiple connectivity options and feature powerful hardware which, along with a rich set of API libraries, allows the developer to write more complex applications.

The two classes largely determine the computational and communication capabilities available on a particular device. While more expensive feature phones may offer better connectivity through Bluetooth or Wi-Fi, processing power remains limited. Smartphones are much less restrictive in

this regard, offering CPUs with frequency in the hundreds of megahertz.[1]

We will now briefly discuss the main platforms and communication channels available on phones today. The selected platform and channel will then serve as a basis for the evaluation and selection of a suitable protocol among those presented in Chapter 3 for implementation on a specific device.

4.1 Mobile Phone Platforms

There is currently a wide range of mobile platforms available on the market. From a functionality perspective, devices from different vendors in both the feature phone and smartphone categories offer similar functionality. The chief differentiating factor among feature phones is thus the set of features offered by the concrete device, while in the smartphone space, devices mostly compete on the hardware level, differentiating themselves with CPU clock speed, memory, or camera and screen resolution.

All major smartphone platforms on the market today contain support for accessing the phone's hardware functionality, such as the various communication channels or sensors available in the device. However, porting an application from one platform to another may be more or less difficult depending on the specific language and toolset required by the platform to build applications.

We will therefore refrain from discussing the technological specifics of individual platforms and instead provide a brief, high-level overview of the most important platforms from a developer's perspective.

Nokia S40 – a feature phone platform used on Nokia phones. The platform does not have its own API for third-party application development; instead, it supports the Java ME environment, allowing for installation of Java MIDlet applications. Access to specific phone functionality, such as messaging, PIM, Bluetooth and others, is facilitated through optional Java ME APIs. The platform does not support multitasking [32].

Symbian/Symbian S60 – a smartphone platform and operating system currently maintained by Nokia. The system uses a realtime microkernel with support for pre-emptive multitasking and memory protection [34]. Symbian supports two main software development languages

1. Devices with dual-core 1+ GHz CPUs, such as the Samsung Galaxy SII or HTC Sensation, were just being introduced on the market at the time of writing this chapter.

and platforms – Java ME and C++.

Older versions of the operating system/platform used a system specific version of C++ called Symbian C++ that contained specific language constructs and deviated from standard C++ [35]. Newer Symbian versions use the Qt SDK along with standard C++ as the main development environment for the platform.

iOS – an operating system developed by Apple and used on its iPhone smartphone and iPad tablet range. The system has its own set of libraries for third-party application development in the Objective-C programming language; other languages are not officially supported.

Applications for iOS are mainly distributed via Apple's App Store, following their submission and review process by Apple. Access to the distribution channel is not free, however, and developers are required to obtain an annual paid membership to be able to sign and distribute their applications.

Android – open source operating system for smartphones and tablets based on the Linux kernel, developed by Google. Android has its own Java-based SDK, with the Java applications running on top of Google's custom Dalvik JVM; this is the primary development environment. It is also possible to compile libraries and applications in other languages to native platform code using either the Android Native Development Kit or through the use of other development tools [28].

The primary distribution channel for Android is Google's Android Market. Unlike Apple's App Store, no paid membership or review process is required to submit applications to the Market, with the only requirement being that the applications are signed with a custom key. Other competing markets, such as the Amazon App Store, also exist that may have different application submission procedures.

Signed applications can also be distributed outside any application marketplaces and installed manually by the user himself.

Windows Phone 7 – operating system for smartphones developed by Microsoft. Third-party application development is facilitated through the XNA framework and Silverlight; the XNA framework, together with a .NET language (officially C#), is used for game development on Windows Phone 7 devices.

BlackBerry OS – operating system used exclusively on the BlackBerry line of smartphones from the Canadian company Research In Motion. Third-party applications are developed in Java ME, with the official SDK containing a set of additional libraries supplied by RIM that provide access to other functionality, such as connectivity, web browser, basic crypto packages and others.

According to Gartner [15], smartphone sales accounted for only 19% of overall mobile phone sales in 2010. According to the same study, Nokia retained its position as the leading vendor in both the overall phone sales as well as in smartphone sales, where Symbian remains the most popular platform with a market share of 37.6%, followed by Android with 22.7%. RIM (BlackBerry) and Apple each held less than 20% of the market share; Microsoft had less than 5% of the overall market share in that period.

With three out of the top five platforms using some version of Java as a development environment, it makes sense to target one of these platforms due to easier portability to the others.

In particular, Java ME implementations allow for almost direct application porting between Symbian and BlackBerry[2], with lightweight implementations being potentially transferable to feature phones that support Java ME as well. Porting to Android would require larger changes at the source code level, however, the basic protocol logic would remain the same and thus only the presentation (graphical) layer and the specifics of the communication layer would have to change.

Our choice of an implementation platform was also restricted by the devices available at our laboratory. These consisted of a number of older Symbian devices, including two Nokia N73s, the Symbian-based Nokia C6-00, and a pair of old Windows Mobile 6 PDAs. This practically limited possible development environments to Symbian C++ and Java ME. Žižkovský has outlined some of the challenges of implementing the basic Burmester-Desmedt protocol in Symbian C++ in [27]. Given the advantages of Java ME outlined above, existence of a good lightweight API for cryptographic and large-integer operations [13] and arguably better code readability, we have decided to implement the selected protocols in Java ME.

2. Unlike Symbian, BlackBerry uses its own set of additional Java libraries to access extra functionality such as Bluetooth. Aside from recompilation, porting an implementation that uses Java ME optional libraries would thus require the specific functionality to be remapped to BlackBerry-specific libraries.

4.2 Communication Channels

Aside from traditional communication channels such as voice or SMS, mobile phones today typically offer a range of data channels as well. There are two key characteristics we require for an efficient implementation of a GKA protocol here – high throughput and low latency. This allows the group members to reliably send and receive messages faster, shortening the duration of individual rounds. Low cost of sending messages also allows for trade-offs between communication and computation to be made on computationally less powerful devices.

Channel confidentiality, on the other hand, is of little importance as all the protocols presented in Chapter 3 are secure against passive adversaries. Active adversaries do, however, incur a communication cost on authenticated GKA protocols, as they can either delay or drop messages, causing the protocol to stall, or send fake messages and thus disrupt the protocol flow completely. No protocol can defend against such denial of service attacks as these are executed on the channel itself. It is therefore desirable to make the communication distance as short as possible, limiting the amount of attackers that can insert themselves between the communicating parties.

There are four data channels that are commonly available on mobile phones today – SMS, MMS, Bluetooth and some form of packet-based communication service over cellular networks used for access to the internet, such as GPRS, EDGE or UMTS. Most modern smartphones also provide IEEE 802.11b/g/n connectivity (Wi-Fi). Following is a brief discussion of characteristics of some of the communicatino channels:

SMS – text messaging service over existing cellular infrastructure. SMS messages are limited to a maximum size of 140 bytes [33]; this amounts to 160 7-bit characters in the default 7-bit encoding, and less in others. Longer text can be sent at once by using multiple messages linked together by segmentation information at the beginning of the message, at the expense of fewer characters in each message.

MMS – multimedia messaging service that allows sending of messages with multimedia content among different devices. Messages are delivered to servers with an HTTP front-end; the recipients are notified about the message via SMS that contains the URL to the data, which can then be retrieved by WAP from the recipient's device or a normal web browser on a PC [31].

Bluetooth – wireless short-range communication technology for personal

area networks in the unlicensed 2.4 GHz radio frequency band. Communication is packet-based, with a master-slave structure where the master's clock defines the intervals in which either the master or its slaves may transmit. Bluetooth also offers authentication and confidentiality services; two devices can become paired when they first start communicating together, establishing a shared link key that can be used to subsequently authenticate the two devices to each other and encrypt communication. Depending on the version supported by hardware, Bluetooth devices can communicate with a throughput of up to 2.1 MBit/s at a typical range of up to 10 m [29].

Wi-Fi – commonly used term for IEEE 802.11a/b/g/n wireless networking technologies certified by the Wi-Fi Alliance. With the exception of 802.11a, Wi-Fi operates in the same 2.4 GHz radio frequency band as Bluetooth but with a higher transmission speed (up to 600 MBit/s in the case of 802.11n [30]). Devices can either communicate directly by forming an ad-hoc network, or connect to an access point that enables access to the internet and local services in the infrastructure mode. Traffic exchanged between the access point and the connected device can be encrypted.

While it is possible to implement a GKA protocol over SMS, MMS or one of the packet-based cellular communication technologies, none of these is particularly suitable for our scenario.

In the case of SMS, protocol messages can be transmitted as the raw bytes of an SMS message. However, unless the underlying application platform allows the developer to programmatically inspect individual bytes of the SMS message, protocol messages have to be encoded either as hexadecimal strings in one of the supported encodings of SMS, or the encoding has to be hardcoded in the protocol implementation to allow SMS messages to be translated to protocol messages. Neither approach is very efficient, with the former allowing only 80 bytes of data to be transmitted per message and the latter making the implementation unnecessarily complicated.

While MMS may allow for larger amount of data to be sent per message, the way recipients handle incoming messages means that an implementation would have to parse the URL for the location of the data from the received SMS and then download the data, instead of allowing the recipient to (passively) receive the data directly. MMS also has the potential to introduce delay into the protocol, as recipients have to wait until the carrier's infrastructure stores the message on a server and notifies the recipient with an SMS, which the recipient then has to parse.

Finally, using a packet-based technology such as GPRS or EDGE, one could implement the protocol over standard TCP/IP. Unfortunately, without fixed addresses and prior associations among members in the group, group members would need to find out their own IP address and manually input the IP address of the other members in the group into the device.

Neither of the three channels are free and members would therefore also have to pay for the transmitted data. Importantly, as the channels send data over the carrier's infrastructure there is no direct "hop-free" link between members of the group. This prevents the implementation from making use of the fact that group members are within close proximity of each other.

Both Bluetooth and Wi-Fi are good candidates for the communication channel used by an implementation of a GKA protocol. Since the Java ME environment does not provide access to the Wi-Fi functionality, we chose to implement protocol communication over Bluetooth.

4.3 Protocol Selection

The specific nature of group communication in a Bluetooth network is an important factor that co-determines the protocol choice. Bluetooth networks (called Piconets) use three-bit active device adressing and thus are limited to 8 active devices in a group. Piconet devices do not have direct links between each other; instead, they share a two-way link with the device that is designated as a master in that network, forming a star topology.

The master's clock determines the communication intervals for both the slaves (devices connected to the master) and itself. While a slave may communicate with the master without restrictions, slave-to-slave communication is facilitated by the master, who has to resend the message to the correct recipient.

In terms of message complexity, this means that any message whose recipient is not the master himself is sent twice – once from the member to the leader, and then from the leader to the correct recipient. Slave-to-slave broadcasts thus have to be transmitted by the master as $n - 2$ messages, each sent to one of the slave devices in the network.

This effectively means that even constant-round protocols will finish in a linear number of rounds in a real-world implementation. It is therefore desirable to minimize the number of broadcasts with respect to the overall number of messages exchanged during the protocol run.

Another important factor is the overall computational complexity of the protocol. While it is desirable to keep the amount of computation as low as

possible, this should not come at the expense of an inordinate increase of communication complexity. A good example of this are the protocols from the Burmester-Desmedt family, such as Dutta-Barua or Katz-Yung. Due to the high number of simultaneous broadcasts in each of the turns, a Bluetooth implementation would require the master to send $O(n^2)$ number of messages in every turn.

With regards to the selection criteria outlined above, we chose to implement a modified version of the Augot et al. protocol, and the unmodified version of the GDH.3 protocol. In addition to good computational complexity, the Augot protocol maps very well to the underlying master-slave topology as members (slaves) only communicate with the leader (master). This amounts to a total of $3n - 3$ messages exchanged during the protocol run.

GDH.3, on the other hand, is slightly less computationally demanding than the Augot protocol at the expense of more messages exchanged during the protocol run. The leader has to resend messages sent from member U_i to U_{i+1} in the original protocol, perform the broadcast for member U_{n-1} and broadcast the final intermediate values in the last round himself. This amounts to a total of $4n - 6$ messages exchanged during the protocol run.

Additionally, we let the leader assign member positions to the slaves in the group, which results to another $n - 1$ messages. We absorb this cost in our implementation of the Augot et al. protocol by including this information in the INIT message which the leader broadcasts in the first round to the rest of the group.

While the low computational complexity of operations performed by regular members in the Boyd-Nieto might be tempting, the required availability of member's public encryption keys makes the protocol unsuitable for our scenario, as members would first have to reliably exchange their encryption keys with the group leader prior to executing the protocol run.

Similarly, we decided against implementing the Dutta-Barua protocol or any other version from the Burmester-Desmedt protocol family – with at most 8 members (including the leader himself), the leader only has to perform 4 more modular exponentiations in the Augot et al. protocol than in Dutta-Barua; however, the linear number of broadcasts in Dutta-Barua results in a quadratic number of messages the leader has to send in a Bluetooth implementation of the protocol. In the authenticated version of the Dutta-Barua protocol, this will also result in a linear number of signature verifications performed by each member. As Jurnečka experimentally verified in [17], digital signatures are by far the most expensive cryptographic operation on mobile phones.

4.4 Protocol Modifications

The base Augot protocol uses digital signatures to sign messages exchanged between the member and the leader, assuming the group members and the leader are each in position of the other's public keys to be able to verify signatures. In practical implementations, this assumes either pre-existing security associations between the members of the group, or the availability of a PKI infrastructure which the members have access to.

In a scenario where a group of users with potentially no personal associations meets ad-hoc and spontaneously decides to agree upon a key, such an assumption seems unrealistic. Users typically will not have a public key with a certificate from a trusted certification authority on their mobile phone, and even if they did, the protocols functionality would rely on everyone in the group having access to the internet to verify the certificate's validity.

A possible solution is to pre-distribute the keys at the beginning of the protocol and store them for later reuse. Here, we could utilize the fact that group members are in close proximity of each other and can therefore visually verify that they received the correct public key. A simple pre-distribution scheme would then be as follows:

$$U_i \rightarrow U_j \quad : \quad PK_i \tag{4.1}$$

$$U_i \leftarrow U_j \quad : \quad msg \text{ received} \tag{4.2}$$

$$U_i \rightarrow^V U_j \quad : \quad h(PK_i) \tag{4.3}$$

$$U_j \quad : \quad \text{If} h(PK_i) = h(msg), \text{accept.} \tag{4.4}$$

The sender would send its public key in plaintext and await confirmation of message delivery from the recipient. At that point, the recipient would not know whether it received the correct key. The sender will then show a hash of the key on the display of his mobile phone, which the recipient would either copy to his own phone for comparison with the hash he computed on the received message, or visually compare the two strings himself to see if they are the same. If so, the key would be stored in the user's mobile phone key database.

Such a solution is still not ideal, however, since the keys themselves do not contain any information on their validity. An adversary could thus use a compromised signing key belonging to a victim with which the user has communicated before to impersonate the victim and either influence or obtain the value of the new group key. The only way to avoid such an attack

is to use the signing keys as one-time keys, performing the predistribution every time a protocol run is started and discarding the keys upon protocol completion.

Since public key signatures are expensive, a better idea would be to get rid of them altogether and instead try to utilize the fact that protocol participants are in close proximity of each other. Authentication thus need not be completely automated and the user can actively participate in the protocol run.

Wong and Stajano proposed the idea of using *auxiliary channels* to facilitate authentication and reduce the amount of expensive computation performed in a protocol run [37]. In addition to the radio frequency channel over which protocol messages are transmitted, uni-directional visual and push-button channels are used along with the RF channel to facilitate message authentication.

To authenticate a protocol message, the sender appends a MAC to the message, computed on the sender's and receiver's identities U_i, U_{i+1}, the original protocol message M_i, and a short random nonce R_i using a random, one-time MAC key K_i as

$$MAC_i = MAC_{K_i}(U_i|U_{i+1}|M_i|R_i).$$

Upon receipt of a message, the receiver would indicate to the sender he is ready to verify the authenticity of the received data (e.g., by pressing an "OK" button on the sender's phone, therefore using the push-button channel).

The sender would then display the random nonce on his screen, which the receiver would copy by sight to his phone. After the recipient successfully entered the random nonce on his device, the sender would send the MAC key that was used to compute the value to the recipient over the radio frequencey channel. Using the key and the random nonce, the recipient can now verify whether the received value is the correct one. If not, the recipient aborts the protocol run. A single round of protocol can be formalized as follows:

$$U_i \rightarrow^{RF} U_{i+1} \quad : \quad M_i|MAC_i \tag{4.5}$$
$$U_i \leftarrow^{PB} U_{i+1} \quad : \quad ACK \tag{4.6}$$
$$U_i \rightarrow^{V} U_{i+1} \quad : \quad R_i \tag{4.7}$$
$$U_i \rightarrow^{RF} U_{i+1} \quad : \quad K_i \tag{4.8}$$
$$U_{i+1} \quad : \quad \text{Verify } MAC_i \tag{4.9}$$
$$U_i \leftarrow^{PB} U_{i+1} \quad : \quad \text{outcome} \tag{4.10}$$

Note that the visual channel, represented by the sender's display on the phone, need not be confidential for the authentication to work – the nonce isn't displayed until the recipient confirms that he has received the message. As the attacker knows neither the nonce nor the MAC key at this stage of the protocol, he is left with the only option of trying to correctly guess the correct key and nonce that was used to produce the MAC on the message. For an HMAC based on SHA-1 along with a 16-bit random nonce, the overall probability of his success is 2^{-176}.

We use the Wong-Stajano authentication scheme as a replacement for the signatures in the Augot protocol, and a means of implementing an authenticated version of the GDH.3 protocol as well.

4.4.1 Message Format Modification

To prevent replay attacks, Augot et al. use nonces together with information from the received message, such as sender and receiver identity, sender nonce and received intermediate values, in messages exchanged during the protocol run to guarantee message freshness. While provably secure, this has the unfortunate side effect of noticeably increasing the size of the message the leader has to transmit in round 3 of the protocol.

By using the authentication scheme outlined above, this information is no longer necessary, since the random nonces and MAC keys guarantee message freshness. An attacker wishing to replay an old message together with its corresponding MAC key would still have to correctly guess the next fresh nonce. The probability of his success can then be further decreased by increasing the size of the nonce.

It is also possible to use timestampts in messages to provide greater guarantees. Messages with timestamps older than a certain threshold would be automatically discarded not allowing the attacker to replay any messages at all.

The final modification to the base Augot protocol is in the init message the leader broadcasts in the first round of the protocol. We instead use the following format:

$$MSG_l^1 = \{0||U_L||U_i||n||size_{MAC}||Mac_l\},$$

where 0 denotes the first round of the protocol, U_L is the leader's identity, U_i is the identity the recipient is to assume for the rest of the protocol, n is the total number of members in the group, including the leader, $size_{MAC}$ is a two-byte value containing the size of the MAC on the message in numbers of bytes, and MAC_l denotes the MAC on the message data in a format as described above. The modified message serves the same purposes as the init message in the original protocol while also allowing us to assign member positions/identities in the group.

Subsequent protocol messages follow the same basic structure, containing the round number, sender and recipient identities, message data size indicator, actual message data, MAC size indicator and the MAC itself. Our implementation of the GDH.3 protocol follows the same structure.

4.5 Implementation details

There are two versions of each protocol we implemented – authenticated and non-authenticated. The core functionality and classes remain the same, and the implementations mostly differ in protocol specific logic.

The application consists of four main parts:

1. GKAMIDlet Application MIDlet class
2. Device and service discovery
3. Main protocol logic
4. Message authentication

The *application MIDlet class* serves as the entry point for the application. It initializes the other parts of the application and displays the main screen, giving the user the option to start the protocol run either as the leader, or as a regular member.

Device and service discovery contains functionality responsible for finding group members wishing to establish the key. This functionality is only executed by the leader.

Main protocol logic contains code for both the leader and member runs of the protocol. Public parameters are instantiated from a custom binary file that contains the generator of the subgroup of quadratic residues in the group of integers modulo a safe prime $\mathbb{Z}_p^*$, as well as the modulus p – this file is located in the Images directory Authenticated versions of the two protocols also have a functional dependency on the *message authentication* part of the protocol which implements the Wong-Stajano authentication protocol. Our implementation uses HMAC based on SHA-1 as the keyed MAC function.

4.5.1 Member Run

Members start the protocol run as slaves. Users operating as members first choose their *distinguishing identifier*, an application specific string we have introduced to allow the leader to easily distinguish among available Bluetooth devices in the area. Member devices then set up their *service record* – a Bluetooth service descriptor that contains identification information about the service/functionality it offers, as well as the URI of the device and other parameters necessar for other devices to connect to it. This allows nearby Bluetooth devices to search for the service offered by that device, and subsequently connect to it. Once done, members will listen for incoming connections from other devices in the area.

Upon successful connection from the leader to a member, both protocol implementations will initiate a group setup phase in which the leader informs the connected members about the total number of protocol participants, and assign the individual members an identifier.

In the Augot protocol, members immediately respond to the initial setup message with their blinded secret. The leader collects the individual contributions, raises the individual values to its secret exponent and sends back the resulting value set. Upon receiving the intermediate values, members then compute the final group key, write the key to an output file in the Image library on the device and display a SHA-1 hash of the group key.

In GDH.3, the leader has to resend messages for the members during the upflow phase, as well as distribute the broadcast from the penultimate member. However, this is completely transparent to the members themselves, who each address the next correct member according to the protocol, while the leader automatically handles delivery of correct messages to the corresponding members.

4.5.2 Leader Run

The leader starts his run searching for potential group members in the area. Bluetooth devices found in the vicinity are then subsequently searched for the corresponding protocol. After the search concludes, the leader's device will display a list of distinguishing identifiers found in the area, allowing the leader to manually select which devices to connect to.

Once the leader has successfully connected to the selected devices, it will assign identifiers to individual group members and start the corresponding protocol run.

Chapter 5

Conclusion

While most group key agreement protocols seem to target large groups with frequent dynamic changes, protocols exist that can be implemented in a mobile phone environment. However, as these protocols typically do not target the platform, practical implementations have to deal with a number of difficulties, such as efficient communication, message size, and authentication.

We have provided an overview of GKA protocols that can be potentially implemented on mobile phones, selected two protocols with respect to the given limitations and characteristics of the implementation platforms, and also presented modifications of the protocols that allow to overcome the difficulties presented by traditional digital signatures as a means of message authentication.

While the authenticated version of the GDH.3 protocol ultimately proved to be less user-friendly than the rest, the authenticated Augot protocol offers security assurance in the presenc of both active and passive adversaries and only requires the regular member to manually authenticate three messages. In places where security against active attacks is not required, the non-authenticated variants can be used, offering faster completion time.

Bibliography

[1] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. Authenticated group key agreement and friends. In *Proceedings of the 5th ACM conference on Computer and communications security*, CCS '98, pages 17–26, New York, NY, USA, 1998. ACM.

[2] Daniel Augot, Raghav Bhaskar, Valerie Issarny, and Daniele Sacchetti. A three round authenticated group key agreement protocol for ad-hoc networks. *Pervasive and Mobile Computing*, 3(1):36 – 52, 2007.

[3] Klaus Becker and Uta Wille. Communication complexity of group key distribution. In *Proceedings of the 5th ACM conference on Computer and communications security*, CCS '98, pages 1–6, New York, NY, USA, 1998. ACM.

[4] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[5] Dan Boneh. The decision diffie-hellman problem. In Joe Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer Berlin / Heidelberg, 1998.

[6] Colin Boyd and Juan Nieto. Round-optimal contributory conference key agreement. In Yvo Desmedt, editor, *Public Key Cryptography PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 161–174. Springer Berlin / Heidelberg, 2002.

[7] Emmanuel Bresson, Olivier Chevassut, Abdelilah Essiari, and David Pointcheval. Mutual authentication and group key agreement for low-power mobile devices. *Computer Communications*, 27(17):1730 – 1737, 2004. Security and Performance in Wireless and Mobile Networks.

[8] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably authenticated group diffie-hellman key exchange – the dynamic case. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 290–309. Springer Berlin / Heidelberg, 2001.

[9] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic group diffie-hellman key exchange under standard assumptions. In Lars Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 321–336. Springer Berlin / Heidelberg, 2002.

[10] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Group diffie-hellman key exchange secure against dictionary attacks. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 603–610. Springer Berlin / Heidelberg, 2002.

[11] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably authenticated group diffie-hellman key exchange. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, CCS '01, pages 255–264, New York, NY, USA, 2001. ACM.

[12] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In Alfredo De Santis, editor, *Advances in Cryptology EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer Berlin / Heidelberg, 1995.

[13] Bouncy Castle. Bouncy castle, 2011. [Online; accessed 29-May-2011].

[14] R. Dutta and R. Barua. Provably secure constant round contributory group key agreement in dynamic setting. In *IEEE Transactions on Information Theory*, pages 2007–2025. 2008.

[15] Gartner. Worldwide mobile device sales, 2011. [Online; accessed 29-May-2011].

[16] I. Ingemarsson, D. Tang, and C. Wong. A conference key distribution system. In *IEEE Transactions on Information Theory*, pages 714–720. 1982.

[17] Filip Jurnečka. *Randomness Extractors in Mobile Devies*. 2010.

[18] Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer Berlin / Heidelberg, 2003.

[19] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. In *IEEE Transactions on Computers*, pages 905–921. 2004.

[20] Mark Manulis. Key agreement for heterogeneous mobile ad-hoc groups. *Parallel and Distributed Systems, International Conference on*, 2:290–294, 2005.

[21] Mark Manulis. Key agreement for heterogeneous mobile ad-hoc groups. *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, 2005.

[22] Adrian Perrig. Efficient collaborative key management protocols for secure autonomous group communication. In *In International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC 99*, pages 192–202, 1999.

[23] Alan T. Sherman and David A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering*, 29:444–458, 2003.

[24] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A secure audio teleconference system. In Shafi Goldwasser, editor, *Advances in Cryptology CRYPTO 88*, volume 403 of *Lecture Notes in Computer Science*, pages 520–528. Springer Berlin / Heidelberg, 1990.

[25] M. Steiner, G. Tsudik, and M. Waidner. Cliques: a new approach to group key agreement. In *1998. Proceedings of the 18th International Conference on Distributed Computing Systems*. 1998.

[26] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM conference on Computer and communications security*, CCS '96, pages 31–37, New York, NY, USA, 1996. ACM.

[27] Jiří Žižkovský. *Generování náhodných sekvencí v mobilních zařízeních*. 2009.

[28] Wikipedia. Android software development, 2011. [Online; accessed 29-May-2011].

[29] Wikipedia. Bluetooth, 2011. [Online; accessed 29-May-2011].

[30] Wikipedia. Ieee 802.11n-2009, 2011. [Online; accessed 29-May-2011].

[31] Wikipedia. Multimedia messaging system, 2011. [Online; accessed 29-May-2011].

[32] Wikipedia. Series 40, 2011. [Online; accessed 29-May-2011].

[33] Wikipedia. Sms, 2011. [Online; accessed 29-May-2011].

[34] Wikipedia. Symbian, 2011. [Online; accessed 29-May-2011].

[35] Wikipedia. Symbian c++, 2011. [Online; accessed 29-May-2011].

[36] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *SIGCOMM Comput. Commun. Rev.*, 28:68–79, October 1998.

[37] F.-L. Wong and F. Stajano. Multi-channel protocols for group key agreement in arbitrary topologies. In *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*. IEEE, 2006.