Etude cryptographique de solution de sécurité pour les environnements distribués sans fil. Application au projet azone.

Raghav Bhaskar

THÈSE

présenteée pour obtenir le grade de

DOCTEUR DE L'ÉCOLE POLYTECHNIQUE

Spécialité :

INFORMATIQUE

par

Raghav BHASKAR

Titre de la thèse :

PROTOCOLES CRYPTOGRAPHIQUE POUR LES RÉSÉAUX MOBILE AD HOC

Soutenue le 26 Juin 2006 devant le jury composé de :

M. Daniel AUGOT (Directeur)
M. Marc GIRAULT (Rapporteur)
Mme Valérie ISSARNY (Directrice)
M. Philippe JACQUET (Président)
M. Refik MOLVA (Rapporteur)
M. Paul MÜHLETHALER (Invité)
M. David POINTCEHVAL (Examinateur)

CRYPTOGRAPHIC PROTOCOLS FOR MOBILE AD HOC NETWORKS

A DISSERTATION
SUBMITTED TO THE DOCTORAL SCHOOL
OF L'ECOLE POLYTECHNIQUE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Raghav Bhaskar
July 2006

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Valérie Issarny Principal Co-Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daniel Augot Principal Co-Advisor

Approved by the Doctoral School of Ecole Polytechnique.

In loving memory of my grandmother
Late Smt. Kaushalya Devi

Abstract

Mobile Ad hoc networks are a step closer to the vision of pervasive computing where all devices dynamically discover each other, organize communication networks between themselves and share resources/information to provide seamless service to the end-user. But providing any reliable service over such a network requires a secure and well-functioning network. Lack of infrastructure, energy-constrained nature of devices and high dynamism in the network makes the task of securing such networks quite challenging.

In this thesis we propose cryptographic protocols, which are a stepping stone to a secure ad hoc network. In particular, we contribute to the areas of key establishment and secure routing in ad hoc networks. Key establishment is concerned with making available cryptographic keys to the devices, necessary for participating in the security services of the network. On the other hand routing needs to be secured in such networks as almost all nodes need to participate in the routing process (for efficiency reasons) and presence of one malicious node could easily have drastic consequences on the routing performance of the whole network. Thus security checks are required to prevent such malicious nodes from hampering the routing process and to recover from it in case they do succeed.

Our first result is a new group key agreement protocol which is especially suitable for ad hoc networks but also outperforms most known protocols for traditional networks as well. The protocol adapts well to the dynamics of the network and is robust enough to deal with message losses and link failures. It requires little self-organization by the nodes in the network. We present some modified versions of the same and

security proofs showing that the security of these protocols is *tightly* related to the security of the Decisional Diffie-Hellman problem. We also discuss issues related to implementation of this protocol in real scenarios.

Our second result is the introduction of the notion of an Aggregate Designated Verifier Signature (ADVS) scheme. An ADVS scheme allows efficient aggregation of multiple signatures on different messages designated to the same verifier. We show how this primitive can be efficiently utilized to secure reactive routing protocols in ad hoc networks. We provide a security model to analyze such schemes and propose an ADVS scheme which aggregates signatures more efficiently than existing schemes.

Acknowledgments

I would like to thank my supervisors Daniel Augot and Valérie Issarny for their constant support and guidance. Thanks are also due to Pascale Charpin, Nicolas Sendrier, Anne Canteaut, Jean-Pierre Tillich and Christelle Guiziou-Cloitre for their help and support. The fruitful discussions with the participants of ACI SERAC deserve special mention, with special thanks to Fabien Laguillaumie, Javier Herranz, Paul Mühlethaler, Cedric Adjih and Caroline Fontaine.

Thanks to Refik Molva and Marc Girault for accepting to be referees for my thesis and David Pointcheval for his valuable time to discuss the security proofs and being in the jury. Thanks to Philippe Jacquet for presiding the jury.

And of course all this would not have been possible without the lively and cheerful atmosphere at work created by all the "stagiares", "doctorants" and "post-doctorants" of Projet CODES.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Ad hoc networks are infrastructure less networks formed by a set of battery-powered mobile devices communicating over a wireless medium. Because of the absence of any fixed infrastructure, all tasks of network set-up and maintenance have to be performed by the nodes themselves. Ad hoc networks where all nodes are in the immediate vicinity (transmission range) of each other are referred to as one-hop networks, while networks where a node has to rely on intermediate nodes to reach other nodes are referred to as multi-hop networks. In the latter, routing is a key functionality which has to be performed by (potentially) all nodes in the network. Since the nodes have a limited energy-source, all functionalities need to be energy-aware, which in turn implies efficiency in both communicational and computational terms. Also battery drain-outs can cause nodes to disappear from the network without notification, causing sudden changes in the composition of the network. Mobility of the nodes is another factor contributing to the fast changing composition of the network. This makes the task of providing a reliable guarantee of quality of service a challenging task. The wireless medium itself is less reliable and a more lossy medium than a wired medium and thus complicates the problem. The relationship between nodes in an ad hoc network is itself quite different from that of nodes in a traditional wired network. In the latter, each single node (at least each edge node) essentially relies on one designated node (the gateway) to communicate with and access services on

other nodes. This designated node is trusted and changes rarely. While in ad hoc networks, nodes have to co-operate with their neighbours (which in turn rely on their neighbours) to access various services. The neighbours of any given node can change often and this change calls for establishment of new trust relationships. Thus there is an inherent need for cooperation and group communication in an ad hoc network.

Ad hoc networks have been long viewed to be of critical value to military operations and battlefield scenarios. The earliest example of a mobile ad hoc network is a packet radio network [JT87]. They are now seen as an attractive alternative to a wired network in many other scenarios including disaster management, vehicular networks, personal area networks and collaborative network applications.

Due to the broadcast nature of the wireless medium, communication in ad hoc networks is highly prone to eavesdropping. Also, lack of any physical access control mechanism permits any node to be part of the network (often without being detected). Low physical security of nodes means that some of the nodes in the network may be potentially compromised as well. Thus the use of cryptographic mechanisms is seen indispensable to achieve secure communication and provide reliable services in such networks. All basic cryptographic mechanisms for achieving confidentiality, integrity, authentication, non-repudiation and availability rely on *cryptographic keys*. Thus making available and maintaining keys (key management) in such dynamic networks with minimal infrastructure support is a basic requirement. The task of key management is complicated by the limited resources (bandwidth, power, computation) available to the nodes as well as lack of any central authority. Providing efficient authentication mechanisms in the absence of any central, trusted authority is a big challenge as well. The need for secure routing in ad hoc networks is much more accentuated than in wired networks because of the participation of almost all the nodes in the routing process. By contrast, in wired networks, the function of routing is performed by a few selected nodes, that are often secured by both physical and cryptographic mechanisms. Thus in brief, the characteristics of ad hoc networks make many existing cryptographic mechanisms (including several authentication and key management protocols) inutile as well as require many other mechanisms to be

modified before they can be employed in such networks.

In this thesis we present the security issues related to mobile ad hoc networks and contribute by proposing new cryptographic primitive and protocols well adapted to such networks. In particular, we concentrate on the two fundamental areas of key establishment and secure routing.

The first contribution of this dissertation is a new group key agreement protocol which allows the participants to agree upon a secret key, which can be used later for other cryptographic mechanisms. The protocol is simple, contributive, robust (to node/link failures) and assumes no information about the topology of the network. We present both unauthenticated and authenticated versions of the protocol along with security proofs in the standard model used for such protocols. We also provide implementation details for employing such a protocol in a typical ad hoc network.

The second contribution of this dissertation is the introduction of the notion of Aggregate Designated Verifier Signatures. These are essentially Designated Verifier Signatures [JSI96] which allow efficient aggregation of multiple such signatures. We provide formal definitions and a model of security for such signatures. We then show how this could be a very useful primitive to authenticate messages in routing protocols (especially reactive routing protocols). These signatures allow authentication of routing messages in ad hoc networks in a more efficient manner than existing schemes.

The thesis is organized as follows: Chapter 2 is an introduction to security issues in Mobile ad hoc network (MANET) scenarios. It includes a short introduction to cryptography for a non-specialist (of cryptography) reader and a short introduction to computer networks meant for a non-specialist (of networks) reader. Section 2.3 discusses the security issues in ad hoc networks and the assumptions about the communication environment made by the schemes presented later on. In Chapter 3 we present a new group key agreement protocol which is especially suitable for ad hoc networks but also outperforms most known protocols in traditional networks as well. We present some modified versions of the same and the security proofs showing that

the security of these protocols is *tightly* related to the security of the Decisional Diffie-Hellman problem. We also discuss issues related to implementation of this protocol in real scenarios. In Chapter 4 we introduce a new cryptographic primitive, which we call an Aggregate Designated Verifier Signature (ADVS) which allows efficient aggregation of multiple signatures designated to the same verifier. We present a concrete instance of the primitive to aggregate multiple signatures intended for a designated verifier, in a way, more efficient than other existing schemes. We also show how this primitive can be very effectively used to secure reactive routing protocols in ad hoc networks. We conclude in Chapter 5 with some insights into areas of future work.

Chapter 2

Introducing MANET Security

In this chapter we explain why security is an important concern for any wireless network and especially for mobile ad hoc networks. We begin with a short and simple introduction to Cryptography. We present details about the paradigm of "provable security" which is used to provide guarantees about the correctness and security of cryptographic primitives and protocols, as it is essential to understand the security proofs presented later in this thesis. That is followed by an introduction to computer networks, meant for a non-specialist reader and details the differences between ad hoc networks and traditional wired networks and the consequent impacts on security issues. Finally we present the areas of ongoing security research in ad hoc networks with focus on areas which this thesis has concentrated on.

2.1 Cryptography

The word cryptography is derived from the Greek word *kryptos*, meaning hidden. Informally, it is the science which deals with transformation of information into a form which is unintelligible to an unintended receiver. Limited use of such techniques has been recorded as early as the 2nd Century B.C. in Egypt. For a good reading on the history of cryptography, we refer the reader to the code book [Sin00]. A more

modern and formal definition of cryptography is:

Definition 2.1 *Cryptography: The study of mathematical techniques related to information security.*

Cryptography can be further viewed as a combination of two disciplines: *Cryptology* and *Cryptanalysis*. *Cryptology* is concerned with the science of developing new techniques related to information security. While *Cryptanalysis* deals with analysis of existing techniques from a point of view of finding weaknesses in them.

2.1.1 Cryptographic Goals

Information security has a large number of objectives. They include ensuring secrecy of information from unauthorized users, endorsement of information by a trusted entity and preventing the denial of previous commitments amongst others. But most of them can be met by four basic goals [MvOV96]: *Confidentiality*, *Data Integrity*, *Authentication* and *Non-repudiation*, which we now describe in detail.

Definition 2.2 *Confidentiality: Keeping the meaning of information away from unintended recipients.*

This is one of the most ancient goals of cryptography. And a whole range of techniques ranging from physical protection to complicated algorithms have been devised to achieve it. Essentially it is achieved by applying a function (called the encryption function) to transform input information (also called **plain text**) to an unintelligible form called the **cipher text**. More formally:

Let $\mathcal{A}$ denote a finite set called the alphabet of definition. For example, $\mathcal{A} = \{0, 1\}$, the binary alphabet, is a frequently used alphabet of definition. $\mathcal{M}$ denotes a set called the message space. $\mathcal{M}$ consists of strings of symbols from an alphabet of definition. An element of $\mathcal{M}$ is called a plain-text message or a plain-text. $\mathcal{C}$ denotes

a set called the cipher-text space. $\mathcal{C}$ consists of strings of symbols from an alphabet of definition, which may differ from the alphabet of definition for $\mathcal{M}$. An element of $\mathcal{C}$ is called a cipher-text. $\mathcal{K}$ denotes a set called the key space. An element of $\mathcal{K}$ is called a **key**. Each element $e \in \mathcal{K}$ uniquely determines a mapping from $\mathcal{M}$ to $\mathcal{C}$, denoted by E_e. E_e is called an **encryption function**. The process of applying the transformation E_e to a message $m \in \mathcal{M}$ is usually referred to as encrypting m. For each $d \in \mathcal{K}$, D_d denotes a mapping from $\mathcal{C}$ to $\mathcal{M}$. D_d is called a **decryption function**. The process of applying the transformation D_d to a cipher-text c is usually referred to as decrypting c or the decryption of c.

Encryption schemes can be divided into two classes: **Symmetric key cryptosystems** and **Asymmetric key cryptosystems**. In symmetric key cryptosystems the decryption function uses the same key as used by the encryption function to transform the message, while in asymmetric systems they are different and it is hard to derive the other if one is known. Examples of the former include DES, AES, IDEA and RC4 [MvOV96, Chapters 6-7] while examples of the latter include RSA, El Gamal encryption and McEliece encryption [MvOV96, Chapter 8].

Definition 2.3 *Data Integrity: Preventing unauthorized alteration (insertion, deletion and substitution) of data.*

Data integrity ensures that the message received by the intended recipient(s) is the same as that sent by the original sender. The most common primitive used to achieve data integrity is a Manipulation Detection Code (MDC). Message Authentication Codes (MACs) provide both data origin and data integrity assurances. Formally, a **Message Authentication Code (MAC)** algorithm is a family of functions M_k parameterized by a secret key k, with the following properties:

- **Ease of computation**: For a given key k and an input x, $M_k(x)$ is easy to compute.
- **Compression**: M_k maps an input x of arbitrary length to an output $M_k(x)$ of fixed length.

- **Computation-Resistance**: Knowing zero or more text - MAC pairs (x_i, $M_k(x_i)$), it is infeasible to compute $M_k(x)$ for some $x \neq x_i$.

Examples of some commonly used MACs include HMAC and DES-CBC [MvOV96, Chapter 9].

Definition 2.4 *Authentication: Providing assurances about the identification of entities or data.*

Authentication is concerned with providing guarantees that entities are who they claim to be or that information is from and as sent by the claimed sender. The former property is referred to as **entity authentication** while the latter is called **data origin authentication** or **message authentication**. A fundamental cryptographic primitive in authentication is a **digital signature**. A digital signature is a mean to bind an entity to a piece of information. More formally, a digital signature scheme can be described as follows:

- Let $\mathcal{M}$ be the set of messages which can be signed. $\mathcal{S}$ is a set of elements called signatures, possibly binary strings of a fixed length.

- S_A is a transformation from the message set $\mathcal{M}$ to the signature set $\mathcal{S}$ and is called a **signing transformation** for entity A. The transformation S_A is kept secret by A and used to create signatures for messages from $\mathcal{M}$.

- V_A is a transformation from the set $\mathcal{M} \times \mathcal{S}$ to the set $\{true, false\}$. V_A is called a **verification transformation** for A's signatures, is publicly known, and is used by other entities to verify signatures created by A.

Examples of well-known digital signature schemes include RSA and DSA signatures [MvOV96, Chapter 11]. Use of digital signatures as well as asymmetric cryptosystems in multi-user scenarios often requires a Public Key Infrastructure (PKI) which helps in establishing trust in the ownership of the public keys.

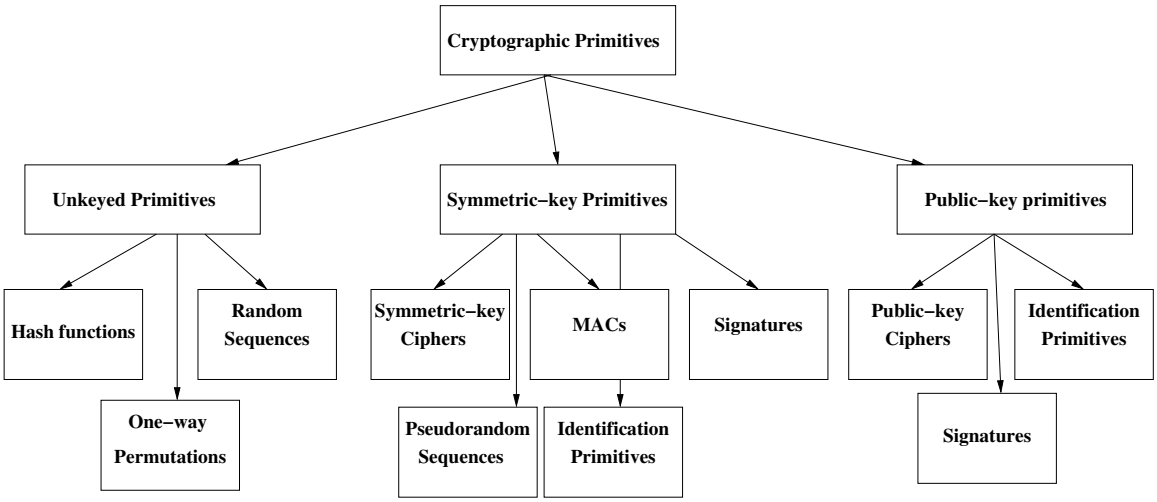


Figure 2.1: Overview of cryptographic primitives [MvOV96]

Definition 2.5 *Non-repudiation: Preventing an entity from denying previous commitments or actions.*

A primitive providing the property of non-repudiation helps in resolving conflicts between parties, one of whom is denying previous actions. A digital signature is one such primitive.

2.1.2 Protocols and Mechanisms

In the previous section we studied the basic goals of cryptography and certain primitives which meet some of these goals. But most real applications may require more than one cryptographic goal to be met in some unique environment. Thus we need a detailed description of the use of multiple cryptographic primitives to achieve a certain security objective in some well-defined environment. Such a detailed description is often referred to as a **protocol**.

Definition 2.6 *Cryptographic protocol: A distributed algorithm, defining precisely the actions and use of primitives by two or more entities to achieve a specific security objective.*

A huge number of protocol objectives exist in literature and for each such objective multiple protocols have been proposed, thus leading to hundreds of protocols. To illustrate how protocol objectives can differ from primitive objectives, we mention below a few protocol objectives and some well known protocols for each objective:

- **Key Exchange**: The aim here is to enable two or more parties to get possession of a shared secret key, which can be used later for some cryptographic purpose. They can be further classified into key distribution (or key transport) and key agreement protocols, and can be realized using both symmetric and asymmetric key primitives. Examples of well-known key transport protocols using symmetric cryptography include Needham-Schroeder [NS78], Kerberos [MNSS87] and Otway-Rees [OR87]. Key transport protocols using asymmetric cryptography include Beller-Yacobi [BY93] and Needham-Schroeder PK [NS78]. Key agreement protocols are mostly based on asymmetric cryptography and include the well-known Diffie-Hellman Key Agreement [DH76], STS [vO93] and MTI [MTI86]. We give below examples of two key exchange protocols to illustrate the differences between various techniques employed to establish keys.

  Table 2.1 specifies a simplified version of the Kerberos protocol, which is key transport protocol for two parties (A and B here) using a trusted server (T). The objective of the protocol is to allow two parties A and B to establish a secret key k, to be used for a specified period of time. Each of them is assumed to share a symmetric key with a trusted server before the start of the protocol. To establish a key with B, A sends a message to the server T with identifiers A and B and a random number (N_A) called a *nonce*. The purpose of the nonce is to uniquely identify each request from A to T to establish a key with B. The server chooses a unique key, k and the period of validity, L of the key. It replies to A's message with two encrypted messages, one meant for A and the other meant for B (called the *ticket*). The message meant for A is encrypted with the shared symmetric key between A and T, K_{AT} and contains the key (to be used by A and B) k, A's nonce, period of validity (lifetime) of the key L and B's identifier. The message meant for B is encrypted with the shared

1. $A \longrightarrow T : \{A, B, N_A\}$
2. $A \longleftarrow T : \{E_{K_{BT}}(k, A, L), E_{K_{AT}}(k, N_A, L, B)\}$
3. $A \longrightarrow B : \{E_{K_{BT}}(k, A, L), E_k(A, T_A)\}$
4. $A \longleftarrow B : \{E_k(T_A)\}$

Table 2.1: Simplified Kerberos key establishment protocol

symmetric key between B and T, K_{BT} and contains the key k, A's identifier and the period of validity (lifetime) of the key L. The third message is from A to B and contains the *ticket* as it is and another quantity (called the *authenticator*) containing A's identifier and the current time, T_A encrypted with the key k. On receiving this message B decrypts the *ticket* to get the key k, A's identifier and the lifetime of the key. If the key is still valid, B decrypts the *authenticator* to check if it contains the correct identifier A and the current time. After these verifications, B replies with A's time-stamp encrypted with the established key k. Thus A is convinced of B' knowledge of the key. This protocol provides both entity authentication and key establishment, with the help of a trusted server. Key transport protocols employing asymmetric cryptography are similar but often do not require a trusted server. Instead they can rely on the other party's public key to securely and efficiently transport the session key. An example of a two party key agreement protocol (with mutual authentication of parties) is shown in Table 2.2. It is inspired from the well-known two party Diffie-Hellman key exchange. The aim of the protocol is to derive a shared key between two parties composed of each one's contribution. Thus the first message is from A to B and is composed of its contribution $g^x \bmod p$ where g is a generator of some group of prime order p and x is randomly chosen from $[1,\ p-1]$. B similarly chooses a random y from $[1,\ p-1]$ and computes $g^y \bmod p$ as its contribution. It computes the shared key $k{=}(g^x)^y \bmod p = g^{xy} \bmod p$. B sends to A its contribution along with an *authenticator* which contains B's signature on both contributions, encrypted with the key k. On receiving this message A computes the key as $k{=}(g^y)^x \bmod p = g^{xy} \bmod p$, decrypts the *authenticator* and verifies B's signature. It then itself sends a similar *authenticator* to B.

1. $A \longrightarrow B : \{g^x \bmod p\}$
2. $A \longleftarrow B : \{g^y \bmod p, E_k(S_B(g^y, g^x))\}$
3. $A \longrightarrow B : \{E_k(S_A(g^x, g^y))\}$

Table 2.2: Station to Station (STS) protocol

Setup(S, n):
(a) Prime $p > max(S, n)$; $a_0 = S$.
(b) $\forall i \in [1, t-1]$, $a_i \xleftarrow{r} [0, p-1]$,
$f(x) = \Sigma_{j=0}^{t-1} a_j x^j$, a random polynomial over $\mathbb{Z}_p$.
(c) $S_i = f(i) \bmod p$, $1 \leq i \leq n$
Secret Computation:
Lagrange Interpolation of t distinct $\{i, S_i\} \longrightarrow a_j$, $1 \leq j \leq t-1$.
$f(0) = a_0 = S$.

Table 2.3: Shamir's (t,n) threshold scheme

- **Secret sharing**: The aim here is to divide a given secret into pieces called shares, which are distributed amongst users in a manner that only pooled shares of specified subsets of users allow reconstruction of the original secret. The most well-known example is Shamir's (t, n) threshold scheme [Sha79], shown in Table 2.3. A trusted server A is given a secret $S > 0$ to distribute among n users such that at least t users have to collude to recover the secret S. The n shares are derived as evaluations of a random polynomial of degree $t-1$ over $\mathbb{Z}_p$ where p is a prime number greater than maximum of S and n. The share (i, S_i) is securely transferred to user u_i. To recover the secret S, any t shares can be used to recover the polynomial (and hence the secret) using Lagrange interpolation.

- **Group Signatures**: Group signatures have the following properties: a) Only members of the group can sign messages, b) The verifier can verify if a given signature is a valid signature from some member of a given group but cannot tell which member of the group actually signed it and c) In case of a dispute, it is possible to "open" the signature (only by the Group Manager) to reveal the

identity of the signer. The first known scheme is from Chaum [Cha].

2.1.3 Attacks and Security Models

A huge number of cryptographic primitives and protocols have been proposed over the years and at the same time, a large number of successful attacks have been mounted on them. By a successful attack is meant the ability of an entity (usually referred to as the *adversary*) to defeat some of the cryptographic objectives of the primitive or the protocol. Such attacks can be broadly classified into two classes:

- **Passive Attacks**: When the adversary merely listens to the communication and uses the monitored information to mount an attack.
- **Active Attacks**: When the adversary not only listens to the communication but also modifies, introduces or deletes messages, or steal identities or uses other such means to mount an attack.

In order to analyze the security of a certain cryptographic primitive or protocol, one needs to define the objective(s) of the protocol and that of the adversary, assumed capabilities of the adversary (the attacker) and present any other assumptions (made about the environment or otherwise). These assumptions form, what is referred to as a security model. Stronger the adversary and fewer the assumptions, stronger is the security model. Below we mention the numerous models (from the strongest to the weakest) in which the security of a cryptographic primitive or a protocol can be analyzed.

- **Unconditional Security**: The strongest measure of security for cryptographic primitives and protocols is unconditional security where the adversary has unlimited computational resources and still cannot defeat a system. One-time pad [MvOV96] is an example of a symmetric encryption achieving unconditional security, but it is unusable in practice.

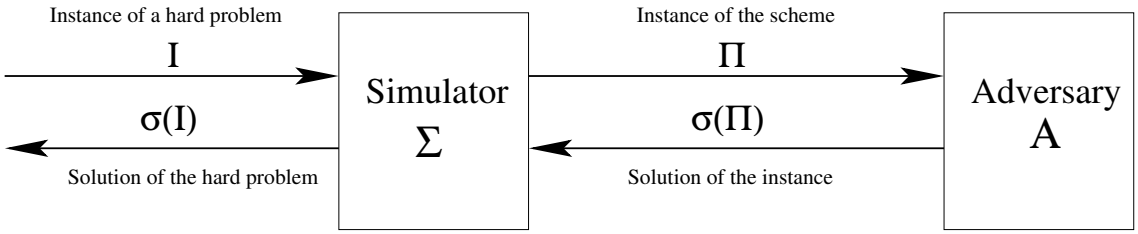


Figure 2.2: Using the adversary to solve a hard problem

- **Complexity theoretic Security**: Here the adversary is considered to be able to mount attacks using space and time polynomial in the security parameter in a certain model of computation. Thus if a system can be shown to resist all such polynomial attacks, it is said to be secure in the complexity theoretic sense.

- **Provable Security**: If the difficulty of attacking a cryptographic system can be shown to be approximately equivalent to that of a well-known hard problem (such as computation of discrete logarithm, integer factorization etc.), the system is said to be provably secure. This is the model employed to prove the security of the new proposed protocols in this thesis and we discuss this model in detail in the following subsection.

- **Computational Security**: In this model, the actual computational effort of implementing the best known attack on a system is compared with the computational resources available to the hypothesized adversary. If the former exceeds the latter, the system is said to be computationally secure.

It is worth noting that there have been many instances of protocols which have been analyzed for their security and considered to be secure, but have been successfully attacked later on. This can happen due to a number of reasons, incorrect assumptions about the adversary's resources, incorrect proofs or discovery of a new efficient algorithm, which helps the attack, are some of them.

Provable Security: This is a model of security, which is increasingly becoming popular to prove the security of both cryptographic algorithms and protocols. The

idea here is to essentially show that the hardness of attacking some cryptographic scheme (protocol or primitive) is computationally equivalent (or tightly related) to that of a well-know hard mathematical problem. When the equivalence of hardness between the two problems is explicitly calculated, it is also often known as **exact security**. The actual proof consists of constructing an efficient algorithm $\mathbf{\Sigma}$, to solve an instance $\mathbf{I}$ of a hard problem, using as a sub-routine, an adversary $\mathbf{A}$, which can solve an instance $\mathbf{\Pi}$ of the scheme, as shown in Figure 2.2. The algorithm $\mathbf{\Sigma}$ is also called a **Simulator** as it has total control on the environment of execution of the cryptographic scheme and controls all interaction that the algorithm $\mathbf{A}$ has with the environment. Thus, in effect, $\mathbf{\Sigma}$ simulates the environment for algorithm $\mathbf{A}$. In doing so, $\mathbf{\Sigma}$ has to ensure that $\mathbf{A}$ cannot distinguish the simulated environment from a real environment with more than negligible probability. The real environment of execution of a cryptographic scheme can consist of various objects including participants, trusted third parties, key set-up functions, other public functions and other objects depending on that particular scheme. The simulator provides access to all these objects via oracles (which it controls). Thus the adversary, in order to interact with these objects, makes queries to the respective oracle and the simulator provides the replies to these queries. As shown in figure 2.3, the adversary interacts with the participants P_1, P_2 ... P_n via queries to the respective oracles, which are controlled by the simulator Σ. Apart from these, access may be available to other scheme dependent oracles, public random oracle (which returns a random number for any new query input) and set-up oracles. Thus any proof of security of a scheme must present a detailed description of the simulator $\mathbf{\Sigma}$ and how it answers all the queries of the adversary $\mathbf{A}$ and finally uses it to solve the given instance $\mathbf{I}$ of the hard problem. One must remember that the simulator is not always required to answer the queries with the correct answer. It can answer wrongly some of the queries of the adversary as long as the adversary has only a negligible probability of knowing that the query was answered wrongly.

An approach often taken to reduce the complexity of the presentation of the proof is to use the so called **Games** approach. Here the interaction between the simulator (now called the **challenger**) and the adversary, where the simulator answers all the queries

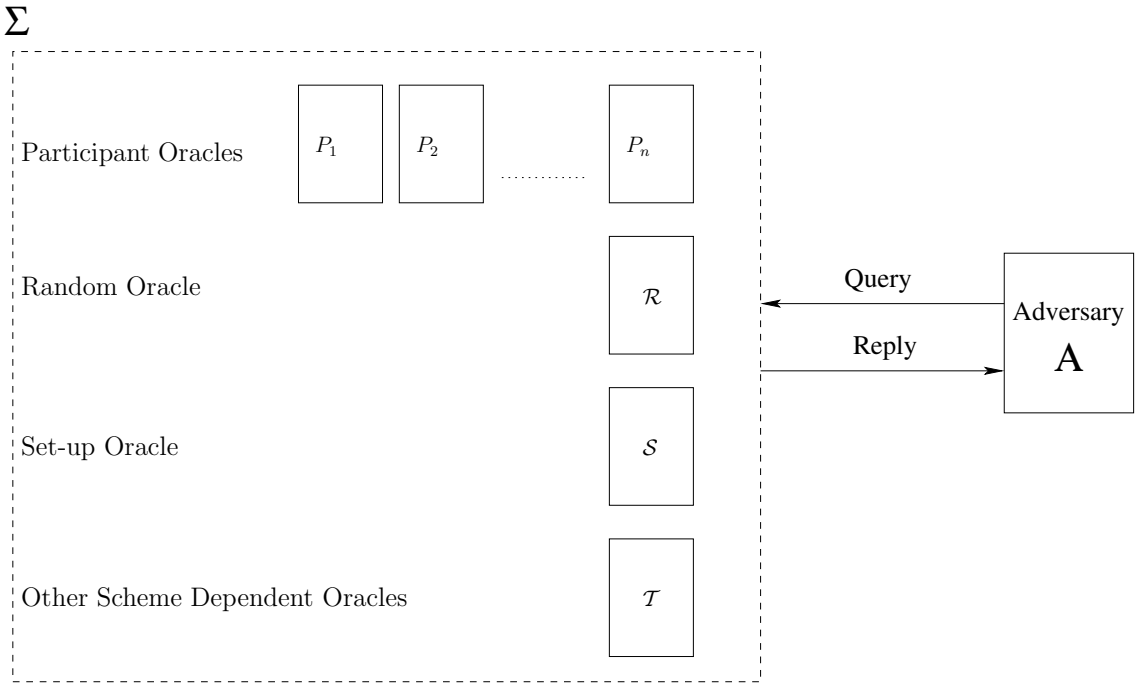


Figure 2.3: The Simulator $\mathbf{\Sigma}$ answers all queries of the adversary

honestly according to the definition of the oracles, is called *Game* 0. Thereafter whenever the challenger deviates from the rules of the original game (for example it changes the way it answers the queries if a certain rare event occurs), this interaction is given another name *Game* i. In doing so, the challenger has to ensure that there is only a negligible difference in the winning probability of the adversary between the two games and thus the success probability of the challenger in solving $\mathbf{I}$ is only affected negligibly. Continuing this way, the challenger stops at a *Game* i, where it is quite easy to calculate the success probability of the adversary in solving the instance $\mathbf{\Pi}$ of the scheme. Thereby relating the probabilities of success of the adversary in any two consecutive games, one can arrive at the probability of success of the adversary in the original game *Game* 0.

We now give an example to illustrate the concept of provable security. We choose a simple example, El Gamal encryption, so as to focus on the proof methodology rather than the actual construction of the proof. El Gamal is an asymmetric encryption cryptosystem proposed by El Gamal in [ElG84] and consists of the following

algorithms:

Setup: it takes as input a security parameter k and outputs a group G of prime order q and a generator g of the group.

KeyGen: it takes as input the group G and its generator g and outputs a public-private key pair (pk, sk) as follows: x in randomly chosen from $\mathbb{Z}_q$. Then ($pk = g^x$, $sk = x$).

Encrypt: it takes as input a message m and a public key $pk = g^x$ and outputs an encryption of m as follows: r is randomly chosen from $\mathbb{Z}_q$. Encryption of m is the pair (y_1, y_2) where $y_1 = g^r$ and $y_2 = m * (g^x)^r$.

Decrypt: it takes as input the cipher text pair (y_1, y_2) and the private key pair $sk = x$ and outputs the plain text as $m = y_2 * (y_1^x)^{-1}$.

To study the security of this encryption cryptosystem, one has to first define the capabilities of an adversary and what it means by a successful attack on the system. Here we show that El Gamal is *semantically secure* against a passive adversary under the Decisional Diffie-Hellman Assumption. Semantic security requires that the adversary is given a public key pk and an encryption e (here (y_1, y_2)) of one of two messages m_0 and m_1 (messages are chosen by the adversary) and the adversary is incapable of telling which of these messages is encrypted. More formally, the adversary plays the following game with some challenger.

Game_0: The challenger runs the Setup and KeyGen algorithms and gives the public key pk thus obtained to the adversary. The adversary chooses two messages m_0 and m_1 and gives them to the adversary. The challenger randomly chooses b from $\{0, 1\}$ and gives the ciphertext (y_1, y_2) of message m_b encrypted using the public key pk. Finally the adversary outputs $b' \in \{0, 1\}$.

If the quantity $|\Pr[b = b'] - \frac{1}{2}|$ is *non-negligible*[1], the adversary is said to have won the game (or successfully attacked the cryptosystem).

[1] A function $\mathbf{v}(n)$ is said to be non-negligible if there exists a polynomial p such that $\mathbf{v}(n) \geq p(n)$ for all sufficiently large n's.

The proof that we present does make some assumptions. Apart from the fact that the adversary is passive, it assumes the Decisional Diffie-Hellman assumption. The Decisional Diffie-Hellman (DDH) Assumption [Bon98] captures the notion that the distributions $X_{DDH} = \{a, b \xleftarrow{r} \mathbb{Z}_q\colon (g, g^a, g^b, g^{ab})\}$ and $X_{Rand} = \{a, b, c \xleftarrow{r} \mathbb{Z}_q\colon (g, g^a, g^b, g^c)\}$ are computationally indistinguishable, where g is a generator of some cyclic group G of prime order q. The advantage of a DDH algorithm D running in time t for G is defined as:

$$\mathrm{Adv}_D(t) = |\Pr[a, b \xleftarrow{r} \mathbb{Z}_q : D(g, g^a, g^b, g^{ab}) = 1] - \Pr[a, b, c \xleftarrow{r} \mathbb{Z}_q : D(g, g^a, g^b, g^c) = 1]|$$

The DDH assumption is the assumption that the advantage of any efficient DDH algorithm is negligible, i.e., $\mathrm{Adv}_{DDH}(t) = \overset{max}{D}\ \mathrm{Adv}_D(t)$ is negligible.

Security Proof: The actual proof is by contradiction. We assume that there is an adversary, A which wins game Game_0 with non-negligible advantage. From such an adversary we construct a DDH algorithm with non-negligible advantage. But this is contradiction to the DDH assumption. Thus we conclude that there exists no adversary who can win game Game_0 with non-negligible advantage.

The DDH algorithm D plays the role of the challenger in game Game_0. It receives as input a DDH challenge tuple $(g,\ g_1,\ g_2,\ g_3)$ from G^4 and introduces it into the game. Instead of running the Setup and KeyGen algorithms to generate the keys, it sets $pk = g_1$ (from the DDH challenge tuple). It gives this to the adversary. The adversary gives two messages m_0 and m_1 to D. D chooses a random bit b from $\{0, 1\}$ and then generates the ciphertext as follows: Instead of choosing a random r from $\mathbb{Z}_q$ to calculate y_1, it sets $y_1 = g_2$ and then sets $y_2 = g_3 * m_b$. It gives the pair $(y_1,\ y_2)$ as the challenge ciphertext to the adversary. If the adversary correctly guesses the value of b i.e. $b' = b$, D returns 1 as its output (implying that the input DDH challenge tuple was indeed a DDH tuple) else it returns 0.

Now it is not difficult to see that if the input DDH challenge tuple is of type (g, g^a, g^b, g^{ab}), the above simulation of D is a perfect simulation of an El Gamal cryptosystem. Thus in this case if the adversary has some advantage in winning

game Game_0, it translates exactly into D's advantage in correctly distinguishing the DDH tuple.

Otherwise if the challenge tuple is of the form (g, g^a, g^b, g^c), adversary can have no advantage in winning game Game_0 as both y_1 and y_2 are uniformly distributed in G irrespective of m_1 and m_2.

Thus $\mathrm{Adv}_{DDH}(t) = \mathrm{Adv}_A(t)$ where $\mathrm{Adv}_A(t) = |\Pr[b = b'] - \frac{1}{2}|$ in Game_0. Note that the run time of the DDH algorithm is almost equal to that of A.

This concludes our basic introduction to cryptography and we now give a small introduction to computer networks with emphasis on ad hoc networks.

2.2 Ad hoc networks

Ad hoc network is a term used to describe a network formed by a set of mobile devices communicating via a wireless medium, in the absence of any fixed facilitator. Such a network is typically set up spontaneously to achieve a specific purpose and thus is limited in both its spatial and temporal extents. It is highly dynamic, as nodes may leave the network due to power shortage (battery run-outs) or wander out of the range of communication of the nearest node. All the basic networking functions like routing, error correction, congestion control etc. have to be performed by the nodes themselves with the cooperation of each other, as there is no fixed infrastructure responsible for providing these services. In the following sections, we present some basic concepts of computer networks followed by a presentation of the key features which distinguish ad hoc networks from conventional networks.

2.2.1 Networking Basics

How a network works can be well understood by understanding the Open System Interconnection (OSI) reference model [osi94]. It gives a conceptual view of the functionalities required to transfer an application message from one node to another

across a network.

The OSI model views each node in the network performing certain functions (required to run the network), which are organized in a layered fashion. There are seven layers of functionality and a layer's contact to its lower layer is only via Service Access Points (SAPs) where it can provide its data (called the service data unit) and expect it to reach its peer layer in the destination machine. Similarly it can receive data from its peer layer on another node via the SAPs of the lower layer on the same node. The actual functioning of the lower layer is transparent to the higher layer. The seven layers of the OSI model, starting from the lowest one, are explained below.

- **Physical Layer**: The physical layer is concerned with the actual transmission of the data over a medium by a node. Thus, it is here where decisions like the voltage to be used for sending a '1' and a '0', the role of each pin in a connector, how to start/tear down a transmission etc. are made. The physical medium to transmit data could be anything from wired mediums like twisted copper pair, fiber optics to wireless mediums like infrared, radio lasers and microwave. Efficient use of the same physical medium for multiple connections' data is done here using various multiplexing techniques.

- **Data Link Layer**: The Data Link Layer (DLL) is concerned with reliable, efficient communication between two nodes connected directly by a communication medium. Some of the key functions performed here are framing of data, error-detection/correction and flow rate control. Thus, in brief, the DLL turns the physical link between two nodes to a reliable efficient communication medium irrespective of the actual reliability of the physical link. In broadcast networks, an additional sub-layer called the Medium Access Control (MAC) layer is required, which deals with the resolution of medium contention issues (see Section 2.2.2 for details).

- **Network Layer**: The network layer is concerned with getting data from one node in the network to another node, multiple hops away, *i.e.*, the basic operation of the subnet. How the messages can be routed in the network efficiently,

congestion-control, quality of service and connection of multiple networks are some of the issues addressed here. Internet Protocol (IP) is an example of a network protocol, and is used on the Internet.

- **Transport Layer**: The transport layer is a true end-to-end layer which is concerned with the functions to be performed at the end hosts. Some of the functions performed are similar to the data link layer. Besides it is concerned with connection set-up/management and multiplexing of multiple applications over a single connection. Transmission Control Protocol (TCP) is an example of a transport protocol, and is used on the Internet.

- **Session Layer**: The session layer enables end users to maintain sessions (dialog handling) between them,thus allowing certain extra services apart from data transfer during the life-time of the session

- **Presentation Layer**: The presentation layer is concerned with the syntax and semantics of the information. Common services include encoding data and converting between different representations used at different machines.

- **Application Layer**: Application layer is where all the user programs (applications) exist. Examples of the latter include http, ftp and smtp.

In figure 2.4 is shown the route taken by a message M from node A to reach the destination node B. The message passes through the seven layers of the end point A, where each layer adds information (called headers) which is relevant to the peer layer of other hosts and is essential for efficient transmission of the message. On intermediate nodes, the message is parsed and modified only up-to the network layer. At the end-point B, the message moves across the seven layers where each layer strips the extra headers added during the route and provides the original message M in the end.

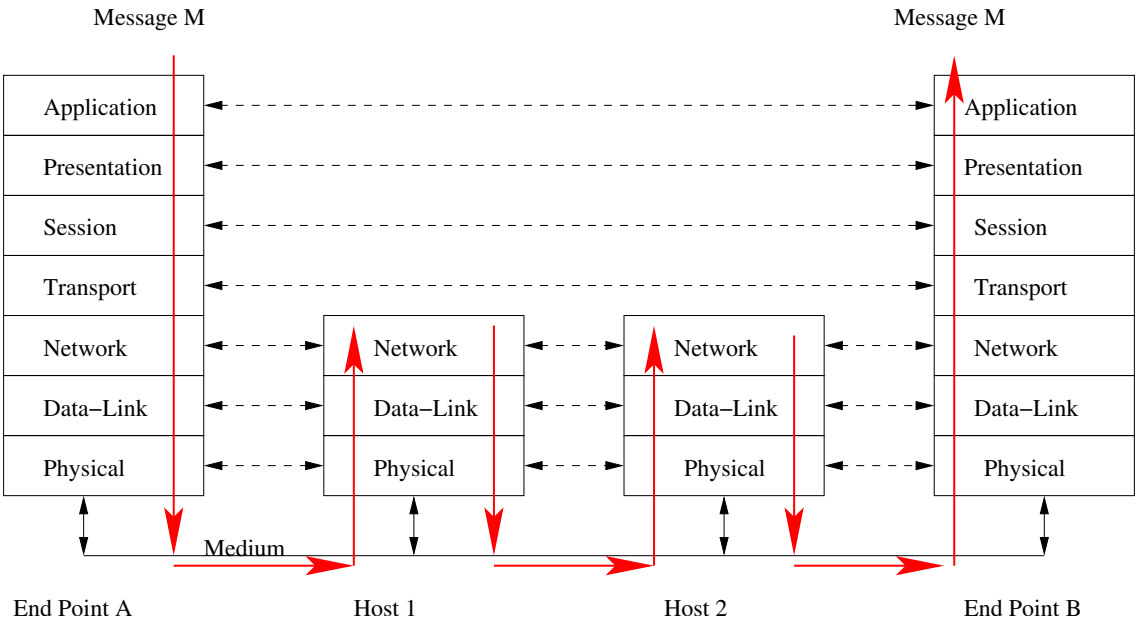


Figure 2.4: The OSI Reference Model

2.2.2 Key Characteristics of Ad hoc Networks

Ad hoc networks have many features, which make them quite distinct from wired networks and thus require innovative ways to implement the network functionalities mentioned in the previous section. We discuss some of the characters which distinguish ad hoc networks below.

- **Wireless medium**: The wireless medium used by the nodes to communicate with each other has time-varying coverage and asymmetric propagation properties. It is less reliable and more prone to interference compared to a wired medium.

- **Mobility**: The nodes in such networks are not expected to be fixed at a certain location, *i.e.*, a fixed destination address does not imply a fixed destination location. Therefore, if a node was reachable in the last transmission does not imply it will be reachable (if at all) in exactly the same fashion the next time around. Thus, single hop networks may require nodes to adjust dynamically

the range of their transmissions while multi-hop networks need to have dynamic routing protocols.

- **Power Management**: As the nodes are not fixed, they rely on batteries as their power source. Thus mechanisms and protocols devised for such networks need to keep the energy constraint in mind.

- **Peer-to-Peer nature**: They are no fixed nodes with pre-defined roles. Thus, all protocols need to be designed for distributed environments composed of "peers" and need to be robust enough to handle these distributed dynamic topologies.

These different characteristics of wireless ad hoc networks require different techniques than wired networks, especially at the three lower-most layers, to effectively perform the network functions. The widely adopted standards for wireless networks, at the physical and data-link layer, include IEEE 802.11 (for wireless local area networks) and IEEE 802.15 (for wireless personal area networks). The 802.11 standard provides MAC sub layer and physical layer specifications for wireless LANs. The 802.15 WPAN work group [802] concentrates on standards for Personal Area Networks or short distance wireless networks. WPANs address wireless networking of portable and mobile computing devices such as PCs, Personal Digital Assistants (PDAs), peripherals, cell phones, pagers and consumer electronics, allowing these devices to communicate and inter-operate with one another. We present the IEEE 802.11 standard, which has taken the lead in wireless ad hoc networks, in detail later. The IETF MANET (Mobile Ad hoc NETworks) working group is looking at efficient routing protocols at the network layer which are lightweight and well adapted to the dynamism in the network. There exist some proposals for transport layer protocols for ad hoc networks as well. In the next few subsections we discuss briefly these proposals.

IEEE 802.11

IEEE 802.11 [i80] is one of the IEEE 802 (Local Area Networks/ Metropolitan Area Networks Overview and Architecture) standards dealing with Wireless LANs (WLANs). It specifies the physical and MAC sub layers for WLANs. The basic topology of a WLAN is called a BSS (Basic Service Set), which is essentially a collection of two or more wireless stations (STAs) that can communicate with each other. If the stations communicate directly with other stations in their coverage area (peer-to-peer communication), the BSS is referred to as an IBSS (Independent BSS) or working in an *ad hoc* mode. Otherwise if all wireless nodes communicate with each other via a fixed access point (AP), the BSS is said to be in *infrastructure* mode. An AP can be considered as a bridge between the wired and wireless networks. The Extended Service Set (ESS) consists of a series of overlapping BSSs connected together by means of a distribution system (DS). This is shown in Figure 2.5. The following are the key features of this standard:

Physical Layer: At the physical layer, WLANs use Radio Frequency (RF) technology in Direct Sequence Spread Spectrum (DSSS) and Frequency Hoping Spread Spectrum (FHSS) modes in the 2.4 GHz (802.11b, 802.11g) range. InfraRed (IR) is another possible medium. This is shown in Figure 2.6. Transmission rates of 54 Mbps are envisioned.

Medium Access Control (MAC): The fundamental access method for WLANs is the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Thus a station senses the medium when it needs to transmit. If the medium is free (for some fixed period[2]), the station transmits. Otherwise if the medium is busy, the station defers transmission till the end of the current transmission, at which point it selects a random back-off interval and transmits at the expiry of this interval if the medium was free during all this time. To further avoid collisions, the transmitting and receiving stations can exchange special control frames (Request To Send (RTS), Clear To Send (CTS)) after determining that the medium is idle and any defferals/back-offs, prior

[2]This is specified in CSMA/CA as the minimum gap between two contiguous frames

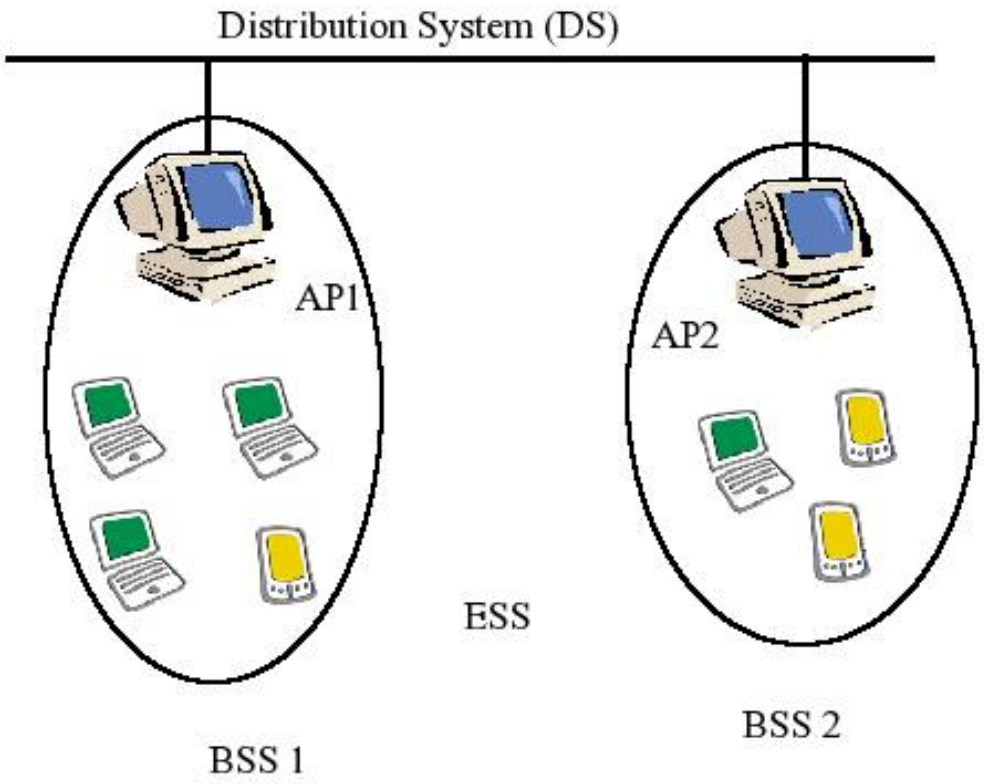


Figure 2.5: IEEE 802.11 Network

802.2			Data−Link
802.11 MAC (CSMA/CA)			
FHSS	DSSS	InfraRed	Physical

Figure 2.6: The IEEE 802.11 Stack

to transmission.

Power Saving: To address the issue of scarcity of power, the standard specifies a mechanism for nodes to go to a sleep mode for long periods of time without losing any data. It is essentially meant for BSS in *infrastructure* mode as it relies on the AP to keep track of "sleeping" stations and store data meant for them.

Security: This is considered an area of fundamental concern for WLANs and thus the standard provides two additional services of Authentication and Privacy. To restrict any arbitrary node from participating in the network, the standard provides shared-key authentication wherein only nodes possessing one common shared key can join the network. Further to prevent unauthorized nodes from eavesdropping, there is a possibility to encrypt all data using the RC4 cipher (in Wired Equivalent Privacy - WEP) or AES (in Wifi Protected Access2 - WPA2). The shared key needs to be placed on each station by a secure channel.

MANET

The objective of the Mobile Ad hoc NETwork (MANET) working group [man] is to standardize IP routing protocol functionality suitable for wireless routing application within both static and dynamic topologies with increased dynamics due to node motion or other factors. Approaches are intended to be relatively lightweight in nature, suitable for multiple hardware and wireless environments, and address scenarios where MANETs are deployed at the edges of an IP infrastructure. Hybrid Mesh infrastructures (e.g., a mixture of fixed and mobile routers) are also meant to be supported by MANET specifications and management features. Routing security requirements and issues are also addressed. The MANET working group is looking into two categories of routing protocols.

- **Reactive Routing Protocols**: Reactive (or on-demand) routing protocols enable dynamic, multi-hop routing between a set of participating nodes. They require messages to be exchanged only when a new (or non-existing) route

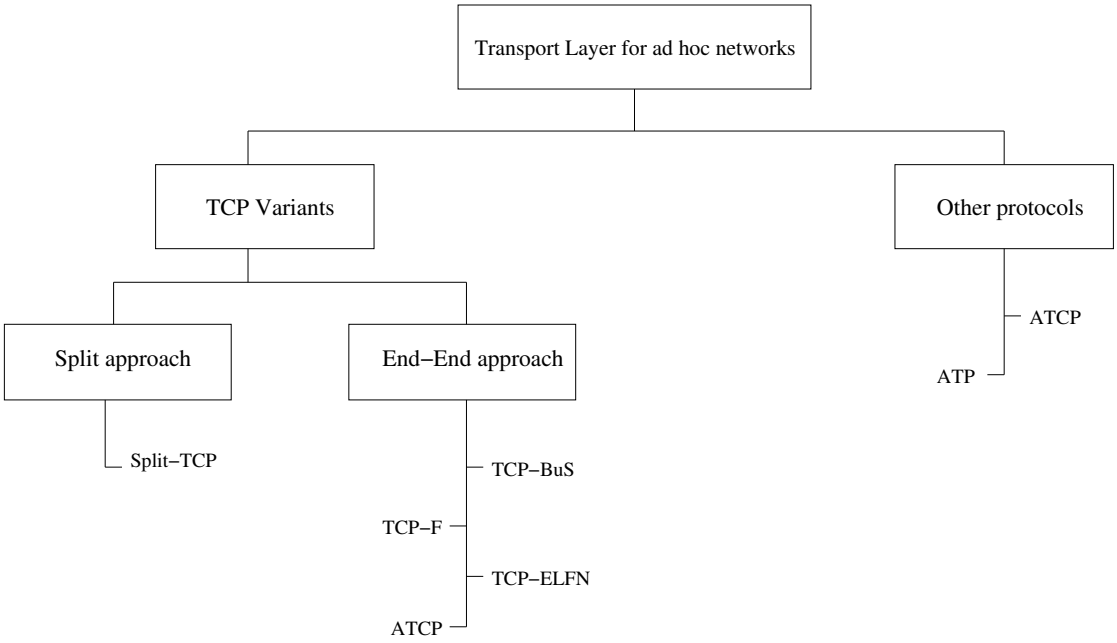


Figure 2.7: Transport Layer protocols for ad hoc networks

between a source and a destination node is to be discovered or when an existing route fails. Dynamic Source Routing (DSR) [dsr], Dynamic MANET On-Demand (DYMO) [dym] are the two internet-drafts in this category.

- **Proactive Routing Protocols**: In contrast, proactive routing protocols require periodic exchange of routing information for loop-free and correct routes. OLSR [ols] is an internet-draft of this category.

The Working Group will also develop a scoped forwarding protocol that can efficiently flood data packets to all participating MANET nodes. The primary purpose of this mechanism is a simplified best effort multicast forwarding function. Simplified Multicast Forwarding for MANET [smf] is an internet-draft in this category.

Transport Layer protocols

The different nature of the communication medium has led to various modifications to the TCP protocol for ad hoc networks as well as proposals for new transport layer

protocols. Examples of the former include TCP-ELFN [HV99], TCP-F [CRVP01], TCP-BuS [KTC01], ATCP [LS01] and Split-TCP [KKFT02] while that of the latter include ACTP [LS99] and ATP [SAHS03]. Split-TCP differs from other TCP variants, as it splits a TCP connection into a set of short concatenated TCP connections (called segments or zones), each having a selected node (known as proxy node) as its terminating point. Local acknowledgments are issued by each proxy node on receiving a packet (which is buffered) from another proxy node or the sender itself. These shorter (effective) path length leads to better throughput. Figure 2.7 classifies them. Please refer to [MM04] for details.

2.3 Securing Ad hoc Networks

In this section we discuss the security issues in ad hoc networks. We present new security issues that are raised due to the different nature of these networks. We also present implications for known security issues in networks. We present in detail two particular issues; that of key establishment and secure routing, which are fundamental for the set-up of other secure services in ad hoc networks and have been the main areas of study in this thesis.

2.3.1 Security issues

In contrast to wired networks, ad hoc networks do not have any physical security of confined offices and buildings or the isolated medium of communication in the form of wires. Thus cryptographic means of security assume an important role. And ensuring secure communication over networks that can be formed in any open place (including forests, battlefields) and use a wireless medium is no easy task. We list below the basic security issues that can be identified for a secure ad hoc network [HBC01, SA99, Sta01].

Access control and Authentication: Controlling who can/cannot enter the network is not a trivial issue. While in traditional wired networks, the secure location of the network access point itself imposes some admission criteria, anything similar is lacking in wireless ad hoc networks. Not only any node with a suitable antenna can receive messages being transmitted over a nearby wireless network, it is also quite simple to introduce messages in the network. Thus in the absence of any admission control procedure, any malicious node can enter the network and play havoc with the functioning of the network. Non-cryptographic solutions to the access control problem are mostly based on trust evaluation and management. Essentially the idea here is that nodes compute a trust level for other nodes based on evidence related to past interactions with them or other nodes' trust levels for that node. Examples of works in this area include [BT04, BEG02]. Cryptographic solutions essentially rely on authentication (mostly using public key cryptography) of nodes and thereafter policies to control access to the network. As establishment of a normal Public Key infrastructure (PKI) is seen infeasible in such networks, varied approaches including distributed PKI schemes [YK03, YK02] as well as PGP like PKI [CBH03] are suggested. For the protocols proposed in this thesis, we need public key authentication mechanisms as well and assume that a distributed PKI like [CBH03] exists.

Key Establishment and Management: Keys are the very basic ingredient of any cryptographic protocol and the protocols of key establishment and management provide a mechanism to establish keys in a newly established network and manage them over an extended period of time. This is one of the focus areas of this thesis. In Section 2.3.2 we explain the problem of key establishment and the traditional approaches taken to resolve it. In Chapter 3 we study the relevance of existing solutions to ad hoc networks and propose a new key agreement protocol especially suited to ad hoc networks.

Secure routing: In multi-hop ad hoc networks, each node possibly may have to aid in the task of routing as there are no dedicated nodes to do this. This also creates the problem that some nodes may disrupt the process of routing. Disruption may be caused by both legitimate nodes (authenticated and authorized nodes) of the network

as well as illegitimate nodes (hidden adversary nodes). The goal of the disrupting node may be to cause failure of packets to reach some particular destination or route certain packets through it or nearby node, among others. It seems impossible to prevent all kinds of attacks in such cooperative networks. But the key goal of secure routing should be to minimize and localize the effect of such attacks and able to come back to normalcy once the disruptive nodes are identified and/or removed. We elaborate the problems with routing in ad hoc networks in Section 2.3.3 and in Chapter 4 review existing proposals and propose a new approach to address this issue.

2.3.2 Key Establishment

The key objective of key establishment is to ensure that all the nodes in the network have the required *keying material* necessary for some of the security services mentioned above. *Keying material* may include public or private keys, any initialization values and any other non-secret parameters. The process of key establishment is not a one time affair, keys may need to be replaced for instance: a) when new nodes enter the network, b) when some nodes leave the network, c) when some nodes get compromised, or d) when the key life-time expires. Thus we need a dynamic key establishment service which can continue to provide the necessary keys during the life-time of the network. Each key has a purpose, period of validity (called its *life-time*) and an ownership set (entities possessing the key). Thus in certain scenarios, we could have a single key for all the nodes in the network during the life-time of the network, while in other scenarios we could have a unique key for each well-defined set of nodes for a well-specified period of time. Keys used by a well-defined set of nodes (we will refer to this set as a group) for a specified period of time is referred to as a *session* key (or group key in this dissertation), as typically it is used to secure a communication session (lasting for some fixed time) between the nodes. Also a particular node may be a member of multiple such (disjoint or overlapping) groups (or sessions) at any given instance of time and thus hold multiple session keys. A key which typically lasts the life-time of a node is often referred to as its *long-term* key. As already mentioned, many key establishment protocols exist in literature. We give

below some formal definitions and discuss the various classes of key establishment protocols.

Key establishment protocols can be broadly classified into two categories, namely: *Key Transport* and *Key Agreement* and can be implemented using symmetric as well as asymmetric cryptographic techniques. We give below formal definitions of Key Transport and Key Agreement and survey such protocols.

Definition 2.7 *Key Transport: A key establishment mechanism wherein one party creates a secret value and then securely transfers it to others.*

Definition 2.8 *Key Agreement: A key establishment mechanism wherein a common secret value is derived by parties from contributions of each and in a way that no one can predetermine the key before the run of the protocol.*

Definition 2.9 *Authenticated Key establishment: A key establishment protocol which provides the property of key authentication, whereby each party is assured that no party aside from identified trusted parties can gain access to the shared secret key.*

Definition 2.10 *Forward Secrecy: A protocol is said to provide forward secrecy if the compromise of the long-term key does not compromise past session keys.*

Key Transport using Symmetric Cryptography: The essential idea behind such protocols is that a trusted third party or one of the members in the group chooses (or obtains) the key and transfers it to all others using symmetric encryption. This assumes some a priori shared long term secret value between the sender and all the recipients. Examples of this class of protocols (for two parties only) include the famous Kerberos protocol [MNSS87], Needham-Schroeder Protocol [NS78] and Otway-Rees [OR87] among others. Such protocols employ symmetric cryptography and are thus computationally efficient in general. The key problem with such protocols for use in ad hoc networks is the requirement of existence of a priori shared secret key between the sender and all others. Establishing $\frac{n(n-1)}{2}$ keys a priori in a group of size n is

quite inefficient. If a single long term secret is shared between all the parties, it leads to loss of *forward secrecy*.

Key Transport using Asymmetric Cryptography: Here, one party chooses a symmetric key and sends it securely to others using public key encryption. If authentication is considered important, signatures may be included as well in the protocol. Computationally, these protocols are more expensive than symmetric key transport protocols. But again revelation of the private key of any party, would lead to the compromise of all past keys sent to it, thus failing to provide *forward secrecy*. Examples include [BY93, NS78].

Key Agreement using Asymmetric Cryptography[3]: The foundational Diffie-Hellman protocol is an example of this class of protocol. Many two-party as well as n-party key agreement protocols have been proposed from it including STS [vO93], MTI [MTI86]. If well implemented, most protocols in this category provide the property of forward secrecy. We discuss in details many existing protocols as well as a new key agreement protocol for n-parties in Chapter 3.

2.3.3 Secure Routing

Due to lack of dedicated fixed infrastructure in ad hoc networks, all nodes have to cooperate with each other to forward one's messages to another. This makes the routing process more prone to attacks as any one of the nodes could misbehave and cause the routing to collapse. Because of mobility it becomes harder to find and isolate the misbehaving node. We precisely define the identified attacks on routing [HP04, MM03].

Cache Poisoning: A malicious node could advertise zero metric for all destination nodes (in Distance Vector Routing protocols) or advertise non-existent links to certain destinations (in Link State routing protocols). This would cause all other nodes to forward packets for all these destinations to the attacker node, which the latter could

[3]The only known example of key agreement using symmetric cryptography is [Blo84] but it requires a central server to pre-distribute keys.

drop and thus disrupt the network routing.

Black hole Attack: A more passive attack is just to drop certain routing messages that a node receives. The dropping of packets could be done in a fashion that suspicions about the presence of a malicious node are not raised. This would make less routing information available to the network to make the routing decisions. Such an attack is difficult to detect as it could be easily confused with a malfunctioning link. Thus the robustness of the protocol is more relevant to handle such attacks, rather than cryptographic mechanisms.

Message Tampering: A node could modify other nodes' routing messages before forwarding. This is quite easy to do if there are no message authentication checks.

Replay attacks: A node could propagate old routing messages (which do not reflect the current topology) in the network to badly affect the routes. This attack works even if there are message authentication techniques being used by the nodes.

Wormhole attacks: If a malicious node X is in the communication range of two nodes who are themselves not in the range of each other, X can simply forward one nodes' messages to other (act as a medium) believing them to think that they are in the direct communication range of each other. And later node X can simply stop this relaying of messages. The attack becomes more significant if this attack is carried between two edge nodes of two different subnets.

Rushing attacks: In reactive routing protocols, if a node can manage to forward route requests (RREQs) to the destination node much earlier than RREQs through other routes (often with the help of another malicious node), it has a probability of forcing routes to pass through it.

The above description of attacks makes it clear that there is a huge range of attacks possible on routing protocols in ad hoc networks. And often it is quite difficult to distinguish an attack from a malfunctioning network (due to link failures, congestion etc.). Thus the goal of secure routing protocols should be to have measures to avoid attacks which affect the whole network severely and to be robust enough to cope with localized and specialized attacks by malicious nodes. In Section 4.1, we present the

state of the art in secure routing protocols for ad hoc networks.

2.3.4 Cryptography in Real Ad hoc networks

Most cryptographic protocols make certain assumptions about the environment of execution of the protocol. In particular, concerning the underlying communication medium, the following assumptions are quite common:

1. Reliable: Messages sent by participant A to participant B are received intact and in order.
2. Synchronization: As most protocols are a sequence of steps often triggered by a receipt of a protocol message, there is an assumption about the ability of the participants to achieve synchronized communication.
3. Broadcast: Protocols relying on a broadcast medium often assume that the same message is received by all nodes.

It is clear that in ad hoc networks such assumptions cannot be made as messages can be lost or arrive out of order or with different modifications for different receivers. Thus the behavior of protocols making such assumptions in ad hoc networks becomes quite unpredictable. While some protocols may simply fail, others can result in more serious security concerns. In the following chapters we propose new cryptographic protocols for ad hoc networks. Initial presentation of the protocols is in tune with standard presentation of similar protocols and the above mentioned assumptions are made. But we present a section on implementation of these protocols in real environments, where we relax these assumptions, leading to slight modifications to the protocols. The security model and proofs of these modified protocols need to be re-examined in such a case.

This concludes our introduction to security issues in ad hoc networks. In the next chapter we explore in detail the area of key establishment in ad hoc networks.

Chapter 3

Key Establishment in Ad hoc Networks

As already discussed, key establishment (management) is a basic cryptographic service needed before any other security service can be deployed in the network. The task of establishing and maintaining cryptographic keys in an ad hoc network is complicated by the absence of any permanent trusted authority and the fast changing composition of the network. Thus there is no single node which can be entrusted with the task of providing keys to other nodes. And as the composition of the network changes often, it is often a desirable property that the keys be changed accordingly, to ensure that only the nodes currently present in the network are able to gain access to the data being exchanged. The keys may also be required to be changed if some node is compromised by an adversary and thus has illegitimate access to some of the keys being used in the network. Thus the following are the requirements desirable from any key establishment mechanism for ad hoc networks:

- **De-centralized**: The key establishment service should not rely on some fixed central trusted node assumed to be present all the time. As connectivity of nodes in an ad hoc network can be severed quite easily, the whole service can

be disrupted if it relies on one single node. A reliable key establishment service, thus, needs to be distributed over several nodes. Due to the peer-to-peer relationship of nodes in an ad hoc network, the keys, preferably, should not be decided by one single entity, but rather, collectively by the nodes.

- **Dynamic**: The key establishment service should be able to handle departure of existing nodes and arrival of new nodes in the network, without requiring a complete re-run of the key establishment protocol. Ideally, new incoming nodes should not be able to gain access to information exchanged in the past while leaving nodes should not be able to gain access to information pertaining to future sessions. Thus it becomes essential that keys are changed to provide key independence across different sessions in the network.

- **Efficient**: The service should be "light" enough so that the nodes spend only a fraction of their time on key-setup functions.

3.1 Background and Existing Work

As seen in Section 2.3.2, key establishment schemes can be broadly classified into three categories. We study here the feasibility of these schemes in ad hoc networks and present the existing literature of relevant ones. In subsequent sections, we provide a new protocol for key establishment (agreement) in ad hoc networks and present a security analysis and implementation details of the same.

A) Key Transport using Symmetric Cryptography: As ad hoc networks lack the presence of an online authority to distribute the keys, most traditional key transport protocols using symmetric cryptography are not well-suited for such networks. Recently, proposals which use only symmetric cryptography in the absence of any permanent authority (a trusted authority is required only before deployment) have been made and use the idea of key pre-distribution [CPS03, DDH$^+$05, EG02]. Such protocols require each node in the network

to be assigned a subset of keys from a common pool of keys before they are actually deployed in the network. The size of the common pool and the subset of keys with each node has to be chosen so as to ensure that with a high probability nodes can find a common key with other nodes. Such solutions are more applicable to sensor networks (can be considered as a special class of ad hoc networks) where all the nodes are deployed by a common authority. In ad hoc networks key pre-distribution mechanisms look much less feasible.

B) **Key Transport using Asymmetric Cryptography**: Traditional mechanisms of key transport using asymmetric cryptography are unsuitable as they require a online trusted authority. In ad hoc networks, key transport using asymmetric cryptography can be achieved by choosing one member as the key server for a certain period of time (session), which chooses a key for the group and and sends it to all the nodes, encrypted using their respective public keys. But the fact that one of the nodes chooses the key for the entire group may not be acceptable in most ad hoc networks which are peer-to-peer networks with only partial trust amongst the members. Also revelation of the secret key of some node compromises all past sessions where this particular node was a participant.

C) **Key Agreement using Asymmetric Cryptography**: Key agreement protocols for groups allow the nodes to cooperate amongst themselves and arrive at a common group key without the need for dedicated trusted servers. These protocols can also adapt to the dynamics of the network by providing efficient means to change the group key as and when required. Thus these protocols seem to meet most of the requirements of a key establishment service for ad hoc networks. Below, we analyze and study the suitability of existing group key agreement schemes for ad hoc networks.

Most group key agreement protocols are derived from the two-party Diffie-Hellman key exchange protocol. GKA protocols, not based on Diffie-Hellman, are few and include [PL00, TT00, BN03]. Both protocols of Li [PL00] and Boyd [BN03] fail to

provide *forward secrecy* while protocol of Tzeng [TT00] is quite resource-intensive and prone to certain attacks [BN03]. Forward Secrecy is a very desirable property for key establishment protocols in ad hoc networks, as some nodes can be easily compromised due to weak physical security of nodes. Thus it is essential that compromise of one single node does not compromise all past session keys. We summarize and compare in Table 3.1 existing GKA protocols based on Diffie-Hellman protocols. We compare essentially the unauthenticated versions of the protocols, as most achieve authentication by using digital signatures and thus have similar costs. We compare the efficiency of these protocols based on the following parameters:

- **Number of synchronous rounds**: In a single synchronous round, multiple independent messages can be sent in the network. Though it has been shown in [BW98] (also see Table 3.2), that at least m messages are required to be exchanged between m participants in a GKA protocol to derive a key composed of each one's contribution, the number of synchronous rounds in which these messages can be exchanged can be much lower if the messages are independent. The total time required to run a round-efficient GKA protocol can be much less than other GKA protocols that have the same number of total messages but more rounds. This is because the nodes spend less time waiting for other messages before sending their own.

- **Number of messages:** This is the total number of messages (unicast or broadcast) exchanged in the network to derive the group key. For multiple-hop ad hoc networks, the distinction between unicast and broadcast messages is important as the latter can be much more energy consuming (for the whole network) than the former.

- **Number of exponentiations**: All Diffie-Hellman based GKA protocols require a number of modular exponentiations to be performed by each participant. A modular operation is a computationally intensive operation compared to other operations and thus gives a good indication of the computational cost for each node.

	Rounds	Messages	Broadcasts	Expo per U_i
ITW [ITW82]	$m-1$	$m(m-1)$	0	m
GDH.1 [STW96]	$2(m-1)$	$2(m-1)$	0	$i+1$
GDH.2 [STW96, BCP02]	m	$m-1$	1	$i+1$
GDH.3 [STW96]	$m+1$	$2m-3$	2	3
Perrig [Per99]	$\log_2 m$	m	$m-2$	$\log_2 m+1$
Dutta [DB05]	$\log_3 m$	m	m	$\log_3 m$
Octopus [BW98]	4	$3m-4$	0	4
BDB [BD94, KY03]	2	$2m$	m	3
BCEP [BCEP03]	2	$2m$	0	$2^{\dagger}$
Catalano [BC04]	2	$2m$	0	$m+1$
NKYW [NLKW04]	2	m	1	$2^{\ddagger}$
KLL [KSML04]	2	$2m$	$2m$	3
STR [SSDW88, KPT04]	2	m	1	i^{*}
Ours [ABIS05, ABISar]	2	m	1	2^{**}

$\dagger$: m exponentiations for the base station.
$\ddagger$: $m+1$ exponentiations and m-1 inverse calculations for the parent node.
$*$: $2m$ exponentiations for the sponsor node.
$**$: m exponentiations for the leader.

Table 3.1: Comparison of GKA protocols

Communication costs still remain the critical factor for choosing energy-efficient protocols for most ad hoc networks. A modular exponentiation can be performed in a few tens of milliseconds on most laptops[1], whereas message propagation in multi-hop ad hoc networks can be easily of the order of few seconds and has energy implications for multiple nodes in the network. As can be seen in Table 3.1, some existing GKA protocols require $O(m)$ rounds of communication for m participants in the protocol. Such protocols do not scale well in ad hoc networks. Even tree-based GKA protocols with $O(\log m)$ rounds can be quite demanding for medium to large sized ad hoc networks. Constant-round protocols are better suited for ad hoc networks. In the following we present these constant round GKA protocols briefly.

Octopus[BW98]: The stepping stone of the Octopus key agreement protocol is a four-party key agreement, which takes place in two rounds with four message exchanges.

[1]See details in Section 3.2.5

Consider four participants M_1, M_2, M_3 and M_4. In Round 1, M_1 and M_2 do a Diffie-Hellman key exchange and agree on $g^{r_1 r_2}$ which we denote by z_{12}. Simultaneously (in the same round) M_3 and M_4 agree on $z_{34} = g^{r_3 r_4}$. In round 2, M_1 and M_3 again do a Diffie-Hellman key exchange using z_{12} and z_{34} to agree on the key $g^{z_{12} z_{34}}$. In the same round M_2 and M_4 can agree on the same key $g^{z_{12} z_{34}}$. To extend the protocol into m users requires partitioning the users into four groups, each having its own controller. The four controllers run the above protocol (amongst themselves) using secrets which include contributions of all their partition members. Clearly this protocol is not easy to implement in ad hoc networks, requiring well defined partitions. Protocols to handle group changes are not proposed in this protocol.

Burmester-Desmedt (BDB in Table 3.1)[BD94, KY03]: m participants in the Burmester-Desmedt protocol are arranged in a ring *s.t.* $M_{m+1} = M_1$. In the first round, each participant M_i sends its contribution g^{r_i} to its left and right neighbors, M_{i-1} and M_{i+1}. In the second round, each participant M_i broadcasts the quantity $X_i = (\frac{g^{r_{i+1}}}{g^{r_{i-1}}})^{r_i}$ where the quantities $g^{r_{i+1}}$ and $g^{r_{i-1}}$ were received from its neighbors. The group key is calculated by participant M_i as follows: $K = (g^{r_i r_{i-1}})^m * X_i^{m-1} * X_{i+1}^{m-2} \ldots * X_{i-2} = g^{r_1 r_2 + r_2 r_3 + \ldots + r_m r_1}$. This ingenious protocol does not adapt well to ad hoc networks. Even if the members could be arranged in a ring based on their increasing identities (IDs), the protocol requires $2m$ broadcasts or m broadcasts and $2m$ unicasts. Also loss of any message in the first round brings the protocol to a halt, as a node requires both its neighbours' messages to broadcast in round 2. There are no proposals for handling group changes in the original proposal.

BCEP[BCEP03]: Bresson *et al.* propose a two round authenticated group key agreement for wireless networks with a fixed base station. The protocol proceeds as follows: Each participant sends its (signed) contribution g^{r_i} to the base station. The Base station raises each received contribution to its (long term) signing key x to get $\alpha_i = g^{r_i x}$. It then increments a counter c (counter value is incremented for each session) and computes $K = H_0(c||\{\alpha_i\}_{i\in[1,m]})$, where H_0 is a hash function and $||$ denotes concatenation. The base station then sends to each participant, M_i, the quantity $K_i = K \bigoplus H_1(c||\alpha_i)$ and the counter value c. Each client checks the counter

and computes $K = K_i \bigoplus H_1(c||\alpha_i)$ and the session key $sk = H(K||ID_g||BS)$, where H_1 and H are hash functions and ID_g is the identifier of the group and BS is the identifier of the base station. The protocol fails to provide forward secrecy if the long-term secret of the base station is revealed.

Catalano[BC04]: The basic functioning of the protocol is as follows. Each participant M_i randomly chooses his contribution a_i (from a group of prime order q) and another random number r_i from $\mathbb{Z}_q$. While a_i is sent to other participants encrypted with their respective public keys, r_i is split using a (m, m) secret sharing scheme between the m participants. Finally the session key is derived from $\prod_i a_i * g^{\sum_i r_i}$. The idea of using r_i is to ensure that the resulting key is random enough (*i.e.*, in rushing scenarios where a participant may choose its a_i after it has seen others a_i's). This is because r_i's are revealed to the participants once they have committed to their a_i and r_i. The protocol is computationally demanding with $O(m)$ exponentiations for each participant. Another drawback is that if any participant's message is lost in first round, the whole protocol is brought to a halt, as the secret sharing schemes implies all m contributions are required to compute $\sum_i r_i$.

STR[SSDW88, KPT04]: This protocol was proposed by Steer*et al.* in [SSDW88] for static groups. Perrig *et al.* proposed procedures to handle group changes in [KPT04]. But this protocol has not been cited as a constant round protocol till now. We explain here in details why this protocol is indeed a constant round protocol. In the first round, each participant M_i broadcasts its contribution g^{r_i} (also known as its blinded key). In the second round, everyone constructs a key-tree as shown in Figure 3.1 where each leaf node represents a participant and the participant with the least g^{r_i} (hereafter called the sponsor) takes the bottom-most, left-most position in the tree. Thereafter, the sponsor broadcasts the set of blinded keys for all the intermediate nodes upto the root node. For the case shown in Figure 3.1, the broadcast message is $\{g^{r_1}, g^{r_2}, g^{r_3}, g^{r_4}, g^{g^{r_1 r_2}}, g^{g^{r_3 \cdot g^{r_1 r_2}}}\}$. The group key is $K = g^{r_4 \cdot g^{r_3 \cdot g^{r_1 r_2}}}$. Participant M_i has to perform $m - i$ exponentiations except the sponsor which has to compute $2m$ exponentiations.

The remaining constant round protocols were proposed later to our work.

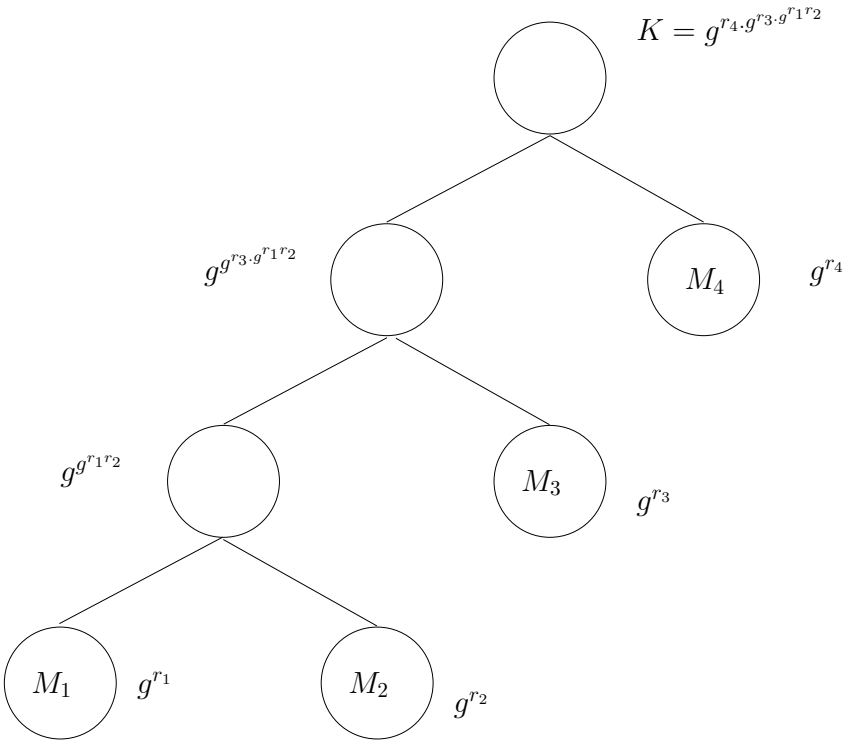


Figure 3.1: The STR Protocol

NKYW[NLKW04]: The original paper proposes this protocol for ad hoc networks composed of devices with unequal computational powers. In the first round, each participant M_i unicasts its contribution $g^{r_i}, i \in [1, n-1]$ to a fixed node (called the parent node) (M_n). The parent node chooses random r and r_n and computes $w = g^r$, $x_n = g^{rr_n}$ and $x_i = (g^{r_i})^r$ for each received g^{r_i}. It broadcasts w and $\{x_n * \Pi_{j \neq i} x_j\}_i$ The key is derived from $\Pi_i x_i$. The protocol does not provide any procedure to handle group composition changes and therefore requires a re-run in that case.

KLL[KSML04]: The basic functioning of the protocol (without specifying the authentication details) is as follows. In round 1, each participant M_i broadcasts its contribution g^{r_i} except participant M_n which also broadcasts $H(k_n||0)$ where H is a hash function and k_n is a random nonce known only to M_n. In Round 2, participant M_i calcultates $T_i = H(g^{r_{i-1}r_i}||I_o||0) \bigoplus H(g^{r_{i+1}r_i}||I_o||0)$ (I_o is the session identifier) and broadcasts k_i (its random nonce) and T_i except participant M_n which broadcasts T_n and $T' = k_n \bigoplus H(g^{r_1 r_n}||I_o||0)$. It can be seen that only the participants can calculate the nonce k_n using T_i's and their contribution. Thus the session key is derived as $H(k_1||k_2 \ldots ||k_n||0)$ where k_n is the only secret. The protocol is quite efficient in both computation as well as communication terms and also has efficient procedures to handle group dynamism. But the protocol suffers from the same drawback as Burmester-Desmedt protocol, *i.e.*, requires the participants to be arranged in a well-defined ring. Non-receipt of the broadcast message (from round 1) of one participant by some participant halts the protocol. That is to say, all broadcasts need to be received by all participants (at least all neighbours in the ring), as they rely on their neighbours messages to compute theirs.

Thus, among the constant round protocols, Octopus [BW98], BDB [KY03] and KLL [KSML04] require special ordering of the participants. This results in messages sent by some participant being dependent on that of others. In such a case, failure of a single node can often halt the protocol. Thus such protocols are not robust enough to adapt well to the dynamism of ad hoc networks. BCEP protocol [BCEP03] fails to provide forward secrecy if the long-term secret of the base station is revealed. Catalano protocol [BC04] is computationally demanding with $O(m)$ exponentiations for each

	Messages	Simple Rounds	Synch. Rounds
Without broadcasts	$2(m-1)$	$\log_2 m$	-
With broadcasts	m	$\log_2 m$	1

Table 3.2: GKA complexity lower bounds [BW98]

participant. Another drawback is that if any participant's message is lost in first round, the whole protocol is brought to a halt, as the secret sharing schemes implies all m contributions are required to compute the key. NKYW [NLKW04] though efficient, does not provide any procedure to handle group composition changes and therefore requires a complete re-run in case of group changes. The STR protocol [SSDW88, KPT04] remains a bit expensive with up to $m-i$ exponentiations for participant M_i and $2m$ exponentiations for the sponsor node (lowermost). The protocol lacks a proof of security against active adversaries. This inability of existing GKA protocols to be efficiently implemented in ad hoc networks was the motivation for our research in this area. An efficient and simple protocol, easy to implement in ad hoc networks was the goal. To that effect, we present a new group key agreement protocol (unauthenticated and authenticated versions) in the following sections with a thorough security analysis and details of implementation. The protocol does not require the participants to be arranged in some fixed order before the protocol starts. In fact the protocol may be initiated by any possible participant (whom we refer to as a *group leader*) and thereafter, group formation and group key calculation can be done simultaneously. Also the unauthenticated version of the protocol meets the lower bound (see Table 3.2) for the number of messages and exceeds the bound for synchronous rounds by one.

3.2 A new Group Key Agreement protocol

We present a new group key agreement protocol derived from a modified version of the well-known two party Diffie-Hellman key exchange in this section. In the following, we use the terms *session* and *group* intermittently but both refer to some sub-set of

nodes[2] in the network, executing the GKA protocol to agree on a secret key. We use the term *group leader* to denote the node which commences the GKA protocol. As our protocol allows any node to start the protocol, any node in the network can be the group leader for a particular session. Our GKA protocol does not impose any constraint on the choice of the group leader and could be chosen randomly, or as well, by the constraints of the overlying application. More details on the group leader election can be found in Section 3.2.5. The protocol is designed for dynamic groups and thus has three sub-protocols namely:

a) **Initial Key Agreement (IKA)** - meant for a group to derive a new group key using everyone's contribution.

b) **Merge** - meant for a group to derive a new group key when new members join the existing group.

c) **Partition** - meant for a group to derive a new group key when some members leave the existing group.

3.2.1 Unauthenticated Version

The protocol, presented here, is unauthenticated and secure against passive adversaries only. We first illustrate the basic principle of key exchange, followed by an explanation of how it is employed to derive Initial Key Agreement (IKA), Join/Merge and Delete/Partition procedures for ad hoc groups.

Notation:
G: A subgroup (of prime order q with generator g) of some group.
$\mathcal{M}$: The set of indices of the participants in a session.
M_i: i^{th} participant amongst the n participants in a session.
M_l: The group leader ($l \in \{1, ...n\}$) in a session.
r_i: A random number (from $[1, q-1]$) generated by participant M_i. Also called the

[2]The term node itself is synonymous with the terms participant and member in this dissertation and they are used intermittently depending on the context.

secret for M_i.
g^{r_i}: The *blinded secret* for M_i (mod q).
$g^{r_i r_l}$: The *blinded response* for M_i (mod q) generated by M_l.
$\mathcal{J}$: The set of indices of the joining participants.
$\mathcal{D}$: The set of indices of the leaving participants.
$x \leftarrow y$: x is assigned y.
$x \stackrel{r}{\leftarrow} \mathcal{S}$: x is randomly drawn from the uniform distribution S.
$M_i \longrightarrow M_j : \{msg\}$: M_i sends message msg to participant M_j.
$M_i \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg\}$: M_i broadcasts message msg to all participants indexed by $\mathcal{M}$.

Protocol Steps:
Round 1: Each M_i responds to the initial request ($INIT$) from M_l, the current group leader (who is randomly chosen), with its blinded secret g^{r_i} to M_l.
Round 2: M_l collects all the received blinded secrets, raises each of them to its secret (r_l) and broadcasts them along with the original contributions to the group *i.e.* it sends $\{g^{r_i}, g^{r_i r_l}\}$ for all $i \in \mathcal{M} \setminus \{l\}$.

Key Calculation: Each M_i checks if its contribution is included correctly and then removes its secret r_i from $g^{r_i r_l}$ to get g^{r_l}. The group key is

$$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}.$$

Note:
1) The original contributions g^{r_i} are included in the last message as they are required for key calculation in case of group modifications (see below).
2) Even though $\Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l}$ is publicly known, it is included in key computation, to derive a key composed of everyone's contribution.

The protocol is formally defined in Table 3.3. We now see how this protocol can be used to derive IKA, Join/Merge and Delete/Partition procedures for ad hoc networks.

Initial Key Agreement - When a group of participants M_i, $i \in [1, n]$ wish to

Round 0
$l \xleftarrow{r} \mathcal{M}, M_l \xrightarrow{B} \mathcal{M} : \{INIT\}$
Round 1
$\forall i \in \mathcal{M} \setminus \{l\}, r_i \xleftarrow{r} [1, q-1], M_i \longrightarrow M_l : \{g^{r_i}\}$
Round 2
$r_l \xleftarrow{r} [1, q-1], M_l \xrightarrow{B} \mathcal{M} : \{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}$
$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 3.3: Initial Key Agreement

Round 0
$\forall i \in \mathcal{J}, r_i \xleftarrow{r} [1, q-1], M_i \xrightarrow{B} \mathcal{M} : \{JOIN, g^{r_i}\}$
Round 1
$r_l \xleftarrow{r} [1, q-1], \mathcal{M} = \mathcal{M} \cup \mathcal{J}, l' \xleftarrow{r} \mathcal{M}, M_l \longrightarrow M_{l'} : \{g^{r_i}\}_{i \in \mathcal{M} \setminus \{l'\}}$
Round 2
$l \leftarrow l', r_l \xleftarrow{r} [1, q-1], M_l \xrightarrow{B} \mathcal{M} : \{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}$
$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 3.4: Join/Merge Protocol

compute a group key, each participant sends its blinded secret g^{r_i} to a randomly chosen group leader M_l ($1 \leq l \leq n$). A single message (Round 2 in Table 3.3) from the group leader, M_l, (using contributions of the other participants) helps each member to compute the group key. An example is provided below.

Suppose M_1 initiates the GKA protocol and 3 members send their blinded secrets, namely, g^{r_2}, g^{r_3} and g^{r_4} respectively. Then the group leader, M_1, broadcasts the following message: $\{g^{r_2}, g^{r_3}, g^{r_4}, (g^{r_2})^{r_1}, (g^{r_3})^{r_1}, (g^{r_4})^{r_1}\}$. On receiving this message, each member can derive g^{r_1} using its respective secret. Thus the key $g^{r_1(1+r_2+r_3+r_4)}$ can be computed.

Join/Merge - Join is quite similar to IKA. Each incoming participant, M_i ($i \in \mathcal{J}$), broadcasts a $JOIN$ request along with its blinded secret, g^{r_i}. The existing group leader (M_l) updates the group composition and chooses a new random secret, r_l,

Round 0

$\forall i \in \mathcal{D}, M_i \xrightarrow{B} \mathcal{M} : \{DELETE\}$

Round 1

$\mathcal{M} = \mathcal{M} \setminus \mathcal{D}, l \xleftarrow{r} \mathcal{M}, r_l \xleftarrow{r} [1, q-1],\ M_l \xrightarrow{B} \mathcal{M} : \{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}$

$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 3.5: Delete/Partition Protocol

and sends all the blinded secrets (including that of the joining members) to the new group leader, $M_{l'}$, (which can be chosen randomly). The new group leader broadcasts a message similar to the round 2 message in IKA *i.e.* all the blinded secrets and the blinded secrets raised to its (new) secret. It is worth noting that the new group leader discards the secret it used during the $JOIN$ request (or secret from last session) and generates a new random secret for the broadcast message. See Table 3.4 for formal specification and below for an example.

Suppose new members, M_9 and M_{10} join the group of M_1, M_2, M_3 and M_4 with their contributions g^{r_9} and $g^{r_{10}}$ respectively. Then the previous group leader (M_1) changes its secret to r_1^* and sends $g^{r_1^*}$, g^{r_2}, g^{r_3}, g^{r_4}, g^{r_9} to M_{10} (say the new group leader). M_{10} generates a new secret r_{10}^* and broadcasts the following message to the group: $\{g^{r_1^*}, g^{r_2}, g^{r_3}, g^{r_4}, g^{r_9}, g^{r_{10}^* r_1^*}, g^{r_{10}^* r_2}, g^{r_{10}^* r_3}, g^{r_{10}^* r_4}, g^{r_{10}^* r_9}\}$. And the new key is $g^{r_{10}^*(1+r_1^*+r_2+r_3+r_4+r_9)}$.

Delete/Partition - When members leave the group, a new group leader is randomly chosen from the remaining members and it changes its secret contribution and sends an IKA Round 2 like message to the group, omitting the leaving member's contribution. Refer to Table 3.5 and below for an example.

Suppose a member, M_2, leaves the group of M_1, M_2, M_3, M_4, M_9 and M_{10}. Then the previous leader, M_{10} changes its secret to $r_{10}^{''}$ and broadcasts $\{g^{r_1^*}, g^{r_3}, g^{r_4}, g^{r_9}, (g^{r_1^*})^{r_{10}^{''}}, (g^{r_3})^{r_{10}^{''}}, (g^{r_4})^{r_{10}^{''}}, (g^{r_9})^{r_{10}^{''}}\}$ to the group. And the new key is $g^{r_{10}^{''}(1+r_1^*+r_3+r_4+r_9)}$.

In the presented protocol, keys across different sessions are kept independent by

requiring at least one member (usually the group leader) to change its secret contribution in each session. Also the role of the group leader can be taken by a different node across different sessions.

3.2.2 Security Analysis

In this section we define the security goals expected to be met by a GKA protocol, explain the security model of Katz-Yung [KY03] (based on the model of [BCP02]) and define other preliminaries before providing a proof of security of our unauthenticated protocol against a passive adversary.

Security Goals - The following security goals can be identified for any GKA protocol.

1) **Key Secrecy**: The key can be computed only by the GKA participants.

2) **Key Independence**: Knowledge of any set of group keys does not lead to the knowledge of any other group key not in this set (see [BM03]).

3) **Forward Secrecy**: Knowledge of some long term secret does not lead to the knowledge of past group keys.

The Security Model - The security model used to provide proof, models interaction of the real participants (modeled as oracles) and an adversary via queries which the adversary makes to the oracles. It is a kind of a "game" between the adversary and the participants, where the adversary makes some queries and finally tries to distinguish a group key from a random quantity for some session he chooses. The model is defined in detail below:

Participants. The set of all potential participants is denoted by $\mathcal{P} = \{U_1, \ldots, U_p\}$ where p is polynomially bounded by the security parameter k. At any given time any subset of $\mathcal{P}$ may be involved in a GKA session. We denote this subset by an index set ($\mathcal{M}$, $\mathcal{J}$ or $\mathcal{D}$) which contains the indices of the session participants with respect to $\mathcal{P}$. Also at any given instant, a participant may be involved (concurrently) in more than

one session. We denote by U_i^s the s^{th} instance of participant U_i and by U^s the s^{th} instance of a generic participant U. Thus one session of the GKA protocol refers to certain instances of certain participants, involved in execution of the GKA protocol and is denoted by $Session = (\mathcal{M}, \mathcal{M}^s)$ where $\mathcal{M}^s$ denotes the respective instances of the participants in $\mathcal{M}$. The group key associated with the instance U^s is denoted by sk_s^U. Before the first protocol run, each participant U runs an algorithm $\mathcal{G}(1^k)$ to generate public/private keys (PK_U, SK_U) which is its *long-term* secret and used for the signature algorithm defined later. Each participant stores its own private key while all the public keys are made available to all participants (and the adversary).

Partners. Partners of an instance U^s are all the instances with which U^s intends to establish the session key. Formally, partnering is defined by the variables session ID (sid_U^s) and partner ID (pid_U^s) of U^s. sid_U^s is protocol-dependent and a function of all messages sent and received by an instance U^s, while pid_U^s is a function of the identities of all the participants in the session.

Correctness. Sessions for which all partnered instances compute the same session key are correct and acceptable sessions. All other sessions are inadmissible and thus rejected.

Adversary. The adversary $\mathcal{A}$ interacts with the participant instances via the following queries:

- Send(U, s, M): This query essentially models the capabilities of an active adversary to send modified/fabricated messages to the participants. The message M is sent to the instance U^s and the reply generated by the instance (in accordance with the protocol) is returned.

As any dynamic GKA protocol P consists of three protocols: **IKA**, **Join** and **Delete**, we define three kinds of **Execute** queries which essentially model the cababilities of a passive adversary.

- Execute$_{ika}$($\mathcal{M}$, $\mathcal{M}^s$): This executes the **IKA** protocol between unused instances of the specified users and returns the transcript of the session.
- Execute$_{join}$($\mathcal{M}$, $\mathcal{M}^s$, $\mathcal{J}$, $\mathcal{J}^s$): This executes the **Join** protocol by adding the users

indexed by $\mathcal{J}$ to an existing group indexed by $\mathcal{M}$ and returns the transcript of the session.

- Execute$_{del}$**(**$\mathcal{M}$**,** $\mathcal{M}^s$**,** $\mathcal{D}$**,** $\mathcal{D}^s$**)**: This executes the **Del** protocol by deleting participants indexed by $\mathcal{D}$ from the existing group indexed by $\mathcal{D}$ and returns the transcript of the session.

- Reveal(U**,** s**)**: This query outputs the session key of instance U^s.

- Corrupt(U**)**: This query outputs the long-term secret (private key) SK_U of participant U.

- Test(U**,** s**)**: This query is allowed only once to the adversary (to be made to a *fresh* instance; see below) during the duration of its execution. A random bit b is generated; if $b = 1$ the session key is returned to the adversary else a random bit string is returned.

Freshness. An instance U^s is *fresh* if both of the following are true: (a) The query **Reveal** has not been made to the instance U^s or any of its partners; (b) No **Corrupt(**U**)** query has been asked to U or any of U^s's partners since the beginning of the game.

Definitions of security. The semantic security of a GKA protocol, P, is tested with the help of a "game" (denoted as $Game_{\mathcal{A},P}$ or in short G_0) which the adversary $\mathcal{A}$ plays with the protocol participants. The goal of the adversary in game G_0 is to correctly guess the bit b used in answering the **Test** query. If $\mathcal{A}$ correctly guesses the bit b, we say that the event Win has occurred. Then the advantage of an adversary $\mathcal{A}$ in attacking the protocol P is defined as $\mathrm{Adv}_{\mathcal{A},P} = 2.\Pr_{\mathcal{A},P}[Win] - 1 = 2.\Pr_{\mathcal{A},P}[b \xleftarrow{r} \{0,1\} : b' = b] - 1$. The maximum advantage over all such adversaries making q queries and operating in time at most t is denoted as $\mathrm{Adv}_P(t, q)$. For a passive adversary we use the notation $\mathrm{Adv}_P(t, q_{ex})$, while for an active adversary we use $\mathrm{Adv}_P(t,\ q_{ex},\ q_s)$, where q_{ex} and q_s denote the number of **Execute** and **Send** queries respectively.

We note that the above model does not address the issue of malicious insiders. Session participants can be corrupted, but only after the session on which the adversary will make the **Test** query has passed. Also our definition of *forward secrecy* does not give access to internal data (any short term secrets or any data stored by the

participant) to the adversary. Only the long term key is revealed. This definition is sometimes referred to as *weak forward secrecy* in literature. Though it may be noted, *strong forward secrecy* can be trivially achieved (albeit at the loss of efficiency) in our protocol by running the **IKA** protocol everytime there is a group membership change. The recently proposed protocol of **KLL** (see Section 3.1) is the only known example (to us) of a dynamic group key agreement protocol achieving *strong forward secrecy*.

Decisional Diffie-Hellman Assumption - The Decisional Diffie-Hellman (DDH) Assumption [Bon98] captures the notion that the distributions $X_{DDH} = \{a, b \stackrel{r}{\leftarrow} \mathbb{Z}_q: (g,\ g^a,\ g^b,\ g^{ab})\}$ and $X_{Rand} = \{a, b, c \stackrel{r}{\leftarrow} \mathbb{Z}_q:\ (g,\ g^a,\ g^b,\ g^c)\}$ are computationally indistinguishable, where g is a generator of some cyclic group G of prime order q. The advantage of a DDH algorithm D running in time t for G is defined as:

$$\mathrm{Adv}_D(t) = |\Pr[a, b \stackrel{r}{\leftarrow} \mathbb{Z}_q : D(g, g^a, g^b, g^{ab}) = 1] - \Pr[a, b, c \stackrel{r}{\leftarrow} \mathbb{Z}_q : D(g, g^a, g^b, g^c) = 1]|$$

The DDH assumption is the assumption that the advantage of any efficient DDH algorithm is negligible, i.e., $\mathrm{Adv}_{DDH}(t) = \max_D \mathrm{Adv}_D(t)$ is negligible.

Secure Signature Scheme - A digital signature scheme $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is defined by a triplet of algorithms:

- $\mathcal{G}$: A probabilistic key generation algorithm which on input 1^k outputs a pair of matching public and private keys (PK, SK).
- $\mathcal{S}$: An algorithm that, given a message m and a key pair $(PK,\ SK)$ as inputs, outputs a signature σ of m.
- $\mathcal{V}$: An algorithm that on input (m, σ, PK), outputs 1 if σ is a valid signature of the message m with respect to PK, and 0 otherwise.

The well accepted security model for a secure signature scheme is EF-CMA (Existential Forgery under Chosen Message Attack)([GMR88]), where the adversary is said to be successful if it produces a valid message-signature $(m,\ \sigma)$ pair for a challenge

public key. In the process it can obtain signatures of any message (except m) of its choice adaptively. We denote by $Succ_{\mathcal{F},\Sigma}(k)$ the probability that an adversary $\mathcal{F}$ succeeds with an existential forgery under adaptive chosen message attack. We say that a signature scheme Σ is secure if $Succ_{\mathcal{F},\Sigma}(k)$ is negligible for any probabilistic polynomial time adversary $\mathcal{F}$. We denote by $Succ_{\Sigma}(t)$ the maximum value of $Succ_{\mathcal{F},\Sigma}(k)$ over all adversaries $\mathcal{F}$ running in at most time t.

Theorem 3.1 *Let P be the protocol as defined above. Let $\mathcal{A}$ be a passive adversary making $q_{ex} = (q_{ika} + q_{join} + q_{delete})$ **Execute** queries to the parties and running in time t. Then Protocol P is a secure GKA protocol. Namely:*

$$\mathrm{Adv}_P(t, q_{ex}) \leq \mathrm{Adv}_{DDH}(t') + \frac{1}{|G|}$$

where $t' \leq t + q_{ex}|\mathcal{P}|(t_{exp} + t_{lkup})$, t_{exp} is the time to perform an exponentiation in G and t_{lkup} is the time to perform a look-up in a table L, to be defined in the proof.

Proof: We show that an adversary who gains an advantage in distinguishing the session key from a random quantity, can be used to build an attacker Δ which gains an advantage in solving an instance of the Decisional Diffie-Hellman (DDH) Problem. The **Send** and **Corrupt** queries are not applicable as we are dealing with a passive adversary and there are no long-term secrets. Thus the only relevant queries are the **Execute**, **Reveal** and **Test** queries. The adversary has access to three kinds of **Execute** queries, namely:

1. **Execute**$_{ika}(\mathcal{M}, \mathcal{M}^s)$: Executes the IKA protocol between the participants indexed by $\mathcal{M}$.

2. **Execute**$_{join}(\mathcal{M}, \mathcal{M}^s, \mathcal{J}, \mathcal{J}^s)$: Executes the Join protocol by adding the participants indexed by $\mathcal{J}$ to $\mathcal{M}$.

3. **Execute**$_{del}(\mathcal{M}, \mathcal{M}^s, \mathcal{D}, \mathcal{D}^s)$: Executes the Delete protocol by deleting the participants indexed by $\mathcal{D}$ to $\mathcal{M}$.

DDH Challenge (g_1, g_2, g_3)	g^{r_i} $(g_2^{\alpha_i} g^{\beta_i})$	$g^{r_i r_l}$ $(g_3^{\alpha_i} g_1^{\beta_i} g_2^{\alpha_i \alpha_l} g^{\beta_i \alpha_l})$
(g^{r_a}, g^{r_b}, $g^{r_a r_b}$)	$g^{r_b \alpha_i + \beta_i}$	$g^{(r_b \alpha_i + \beta_i)(r_a + \alpha_l)}$
(g^{r_a}, g^{r_b}, g^{r_c})	$g^{r_b \alpha_i + \beta_i}$	$g^{(r_c - r_a r_b)\alpha_i + (r_b \alpha_i + \beta_i)(r_a + \alpha_l)}$

Table 3.6: Blinded secrets and responses from DDH Challenge

Assume the adversary $\mathcal{A}$ distinguishes the session key with a probability non negligibly greater than 0.5. We construct from $\mathcal{A}$ a DDH algorithm Δ that receives as input an instance $D = \{g, g_1, g_2, g_3\}$ and predicts if it is an instance from $(g, g^{r_a}, g^{r_b}, g^{r_a r_b})$ or $(g, g^{r_a}, g^{r_b}, g^{r_c})$ with a non-negligible advantage.

Δ uses the instance D to derive the replies to the **Execute** queries of the adversary. All blinded secrets generated by Δ to answer $\mathcal{A}$'s queries are generated as follows: Δ picks two random exponents α_i and β_i from $[1, q-1]$ and computes $g_2^{\alpha_i} g^{\beta_i}$ (g_2 is from the given instance D), which is used as the blinded secret g^{r_i}. The corresponding blinded response ($g^{r_i r_l}$) is generated as follows : $g_3^{\alpha_i} g_1^{\beta_i} g_2^{\alpha_i \alpha_l} g^{\beta_i \alpha_l}$ where g_1 and g_3 are from the instance D and α_l is the (random) secret of the group leader of that session. Δ stores g^{r_i}, α_i and β_i as a table entry in table L. It sets c (the operation counter) and L to 1 and null respectively. Table 3.6 shows the relation between the blinded secrets and blinded responses as a function of the input DDH challenge. Thus the blinded response is a Diffie-Hellman value of $g^{r_b \alpha_i + \beta_i}$ and $g^{r_a + \alpha_l}$ if g_3 equals $g^{r_a r_b}$ else randomly distributed in G.

Let us present in detail how Δ answers $\mathcal{A}$'s queries.

1) **Execute**$_{ika}$($\mathcal{M}$, $\mathcal{M}^s$)

Δ randomly chooses l from $\mathcal{M}$ and generates random numbers α_i and β_i, $\forall i \in \mathcal{M}$. It uses the values $g_2^{\alpha_i} g^{\beta_i}$ and $g_3^{\alpha_i} g_1^{\beta_i} g_2^{\alpha_i \alpha_l} g^{\beta_i \alpha_l}$ as g^{r_i} and $g^{r_i r_l}$, $\forall i \in \mathcal{M} \setminus \{l\}$ and returns the set $\{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}$ as the query response.

It sets $c = c + 1$, $L = L \cup \{(g^{r_i}, \alpha_i, \beta_i)_{i \in \mathcal{M}}\}$ and $State(\mathcal{M}, \mathcal{M}^s) = \{leader = l, \{g^{r_i}\}_{i \in \mathcal{M}}, Key = g_1 g^{\alpha_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l}\}$.

2) **Execute**$_{join}$($\mathcal{M}$, $\mathcal{M}^s$, $\mathcal{J}$, $\mathcal{J}^s$)

Δ generates random numbers α_i and β_i and uses the values $g_2^{\alpha_i} g^{\beta_i}$ as g^{r_i}, $\forall i \in \mathcal{J}$. It also generates new random contributions, α_{leader} and β_{leader} for the old leader and uses $g_2^{\alpha_{leader}} g^{\beta_{leader}}$ as $g^{r_{leader}}$. It chooses a random l from $\mathcal{M} \cup \mathcal{J}$, and uses the values $g_3^{\alpha_i} g_1^{\beta_i} g_2^{\alpha_i \alpha_l} g^{\beta_i \alpha_l}$ as $g^{r_i r_l}$, $\forall i \in \mathcal{M} \cup \mathcal{J} \setminus \{l\}$, where α_i and β_i are the corresponding entries of g^{r_i} retrieved from table L. The query response is $\{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \cup \mathcal{J} \setminus \{l\}}$.

It updates $c = c + 1$, $L = L \cup \{(g^{r_i}, \alpha_i, \beta_i)_{i \in \mathcal{J}}\} \cup (g^{r_{leader}}, \alpha_{leader}, \beta_{leader})$, $\mathcal{M} = \mathcal{M} \cup \mathcal{J}$ and $State(\mathcal{M}, \mathcal{M}^s) = \{leader = l, \{g^{r_i}\}_{i \in \mathcal{M}}, Key = g_1 g^{\alpha_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l}\}$.

3) **Execute**$_{del}(\mathcal{M}, \mathcal{M}^s, \mathcal{D}, \mathcal{D}^s)$

Δ generates new random contributions, α_{leader} and β_{leader} for the old leader and uses the value $g_2^{\alpha_{leader}} g^{\beta_{leader}}$ as $g^{r_{leader}}$. Then Δ chooses l at random from $\mathcal{M} \setminus \mathcal{D}$. It uses the values $g_3^{\alpha_i} g_1^{\beta_i} g_2^{\alpha_i \alpha_l} g^{\beta_i \alpha_l}$ as $g^{r_i r_l}$ and generates the query response as $\{g^{r_i}, g^{r_i r_l}\}$, $\forall i \in \mathcal{M} \setminus \mathcal{D} \setminus \{l\}$, where again α_i and β_i are the corresponding entry of g^{r_i} retrieved from table L.

It updates $c = c + 1$, $L = L \cup (g^{r_{leader}}, \alpha_{leader}, \beta_{leader})$, $\mathcal{M} = \mathcal{M} \setminus \mathcal{D}$ and $State(\mathcal{M}, \mathcal{M}^s) = \{leader = l, \{g^{r_i}\}_{i \in \mathcal{M}}, Key = g_1 g^{\alpha_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l}\}$.

4) **Reveal**(U, s)

Δ retrieves the variable $State(\mathcal{M}, \mathcal{M}^s)$ to which the participant U^s belongs and answers with the value contained in Key.

5) **Test**(U, s)

Δ answers exactly as the **Reveal** query, i.e., it retrieves the variable $State(\mathcal{M}, \mathcal{M}^s)$ to which the participant U^s belongs and answers with the value contained in Key.

Finally, Δ outputs the same bit as the adversary $\mathcal{A}$. Now, we know from the random self-reducibility of Decisional Diffie Hellman [BBM00] that if the DDH input instance (g, g_1, g_2, g_3) is indeed of the form $(g, g^{r_a}, g^{r_b}, g^{r_a r_b})$, all randomly generated tuples by Δ are also valid DDH tuples. And if the given instance is of the form $(g, g^{r_a}, g^{r_b}, g^{r_c})$ $(r_c \neq r_a * r_b)$, all randomly generated tuples are random and uniformly distributed. But in case r_c happens to be equal to $r_a * r_b$ (which happens with probability $\frac{1}{|G|}$),

the randomly generated tuples will be valid DDH tuples. Thus for all but probability $\frac{1}{|G|}$, Δ distinguishes the DDH challenge correctly when the adversary distinguishes the session key correctly. Thus[3] $\text{Adv}_{DDH}(t') \geq \text{Adv}_P(t, q_{ex})$ - $\frac{1}{|G|}$.

The running time of Δ is bounded by the running time of $\mathcal{A}$ and the time to perform at most $|\mathcal{P}|$ exponentiations plus the time to look up in table L for at most $|\mathcal{P}|$ exponent entries during q_{ex} queries i.e. $t' \leq t + |\mathcal{P}| * q_{ex} * t_{exp} + |\mathcal{P}| * q_{ex} * t_{lkup}$.

Thus, we get the desired result, $\text{Adv}_P(t, q_{ex}) \leq \text{Adv}_{DDH}(t + q_{ex}|\mathcal{P}|(t_{exp} + |t_{lkup})) + \frac{1}{|G|}$.

3.2.3 Authenticated Version

The protocol presented above lacks authentication of messages and is secure against passive adversaries only. We now show how to convert this protocol secure against active adversaries. One possibility is to use the authentication "compiler" of Katz-Yung proposed in [KY03] which transforms any GKA protocol P, secure against passive adversary, to an authenticated GKA protocol P', secure against an active adversary. It achieves this by enhancing the protocol to include a (pre-)round where everyone broadcasts its identity and a random nonce. Thereafter each message is accompanied by a signature on the message, identities of the participants and their nonces. More formally, given a protocol P, secure against passive adversary, the compiler constructs protocol P' as follows:

1. During the initialization phase, each party $U \in \mathcal{P}$ generates the verification/ signing keys (PK'_U, SK'_U) by running $\mathcal{G}(1^k)$. This is in addition to any keys (PK_U, SK_U) needed as part of the initialization phase for P. In our case $(PK'_U, SK'_U) = (PK_U, SK_U)$.

2. Let $U_1, ..., U_n$ be the identities (in lexicographic order) of users wishing to establish a common key, and let $\mathcal{U} = U_1|...|U_n$. Each user U_i begins by choosing a random nonce $r_i \in \{0,1\}^k$ and broadcasting $U_i|0|r_i$. After receiving the initial

[3] A more formal derivation of the result is presented in Section 3.2.4

broadcast message from all other parties, each instance stores $\mathcal{U}$ and $r_1|...|r_n$ as part of its state information. This adds a round to the existing protocol.

3. The members of the group now execute P with the following changes:

   - Whenever instance U_i^s is supposed to broadcast $U|j|m$ as part of protocol P (j - sequence number of message m from U), the instance instead signs $j|m|\mathcal{U}|r_1|...|r_n$ using SK'_U to obtain signature σ, and then broadcasts $U|j|m|\sigma$.
   - When instance U_i^s receives message $V|j|m|\sigma$, it checks that: (1) $V \in pid_U^s$, (2) j is the next expected sequence number for messages from V , and (3) (using PK'_V) σ is a correct signature of V on $j|m|\mathcal{U}|r_1|...|r_n$. If any of these are untrue, U_i^s aborts the protocol. Otherwise, U_i^s continues as it would in P upon receiving message $V|j|m$.

4. Each non-aborted instance computes the session key as in P.

Then it is proved that if P is a secure GKA protocol, then the protocol P' is a secure Authenticated GKA protocol. Namely,

Theorem 3.2 ([KY03]) $\mathrm{Adv}_{P'}(t, q_{ex}, q_s) \leq \frac{q_s}{2} * \mathrm{Adv}_P(t', 1) + \mathrm{Adv}_P(t', q_{ex}) + |\mathcal{P}| * Succ_\Sigma(t') + \frac{q_s^2 + q_{ex} q_s}{2^k}$
where $t' = t + (|\mathcal{P}|q_{ex} + q_s).t_{P'}$, $t_{P'}$ *is the time to execute* P', q_{ex} *and* q_s *are the number of* ***Execute*** *and* ***Send*** *queries respectively and* Σ *is a secure signature scheme and* k *is a security parameter (see [KY03] for proof details).*

Thus applying the above compiler to our protocol yields a 3-round authenticated GKA protocol, P' with the following security reduction:

Theorem 3.3 $\mathrm{Adv}_{P'}(t, q_{ex}, q_s) \leq \frac{(q_s+2)}{2} * \mathrm{Adv}_{DDH}(t') + |\mathcal{P}| * Succ_\Sigma(t') + \frac{q_s^2 + q_{ex} q_s}{2^k} + \frac{q_s}{2|G|}$.

But this degrades the advantage in solving the DDH instance by a factor of $O(q_s)$ from the advantage of the active adversary. Also the new round where everyone broadcast

its identity and a random nonce, makes the protocol less efficient. Thus we propose a new mechanism to achieve authentication in our protocol, which neither requires a round of n broadcasts nor degrades the security reduction.

We propose below an authenticated version of our group key agreement protocol. The setting and notation is as in Section 3.2.1.

Please note that in the following rounds each message is digitally signed by the sender (σ_i^j is signature on message msg_i^j in Tables 3.7-3.9) and is verified (along with the nonces) by the receiver before following the protocol.

Protocol Steps:

Round 1: The chosen group leader, M_l makes a signed initial request ($INIT$) with his identity, U_l and a random nonce N_l to the group $\mathcal{M}$ (see Table 3.7 for exact message contents).

Round 2: Each interested M_i responds to the $INIT$ request, with his identity U_i, a nonce N_i and a blinded secret g^{r_i} to M_l (see Table 3.7 for exact message contents).

Round 3: M_l collects all the received blinded secrets, raises each of them to its secret (r_l) and broadcasts them along with the original contributions to the group, i.e. it sends $\{U_i, N_i, g^{r_i}, g^{r_i r_l}\}$ for all $i \in \mathcal{M} \setminus \{l\}$.

Key Calculation: Each M_i checks if its contribution and nonce is included correctly and obtains g^{r_l} by computing $(g^{r_i r_l})^{r_i^{-1}}$. The group key is

$$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}.$$

The protocol is formally defined in Table 3.7. We now see how this protocol can be used to derive IKA, Join/Merge and Delete/Partition procedures for ad hoc networks.

Initial Key Agreement - Thus, blinded secrets, g^{r_i}'s, of all potential members, U_i's, are collected by the group leader. A single message (Round 3 in Table 3.7) from the group leader, U_l, (using contributions of the joining participants) helps each participant to compute the group key.

Round 1
$l \xleftarrow{r} \mathcal{M}, N_l \xleftarrow{r} \{0,1\}^k$
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^1 = \{INIT, U_l, N_l\}, \sigma_l^1\}$
Round 2
$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_l}\{msg_l^1, \sigma_l\} == 1)$, $r_i \xleftarrow{r} [1, q-1], N_i \xleftarrow{r} \{0,1\}^k$,
$U_i \longrightarrow U_l : \{msg_i = \{IREPLY, U_l, N_l, U_i, N_i, g^{r_i}\}, \sigma_i\}$
Round 3
$r_l \xleftarrow{r} [1, q-1]$,
$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)$ and N_l is as contributed
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^2 = \{IGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M}\setminus\{l\}}\}, \sigma_l^2\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} and N_i are as contributed
$Key = g^{r_l(1+\sum_{i \in \mathcal{M}\setminus\{l\}} r_i)}$

Table 3.7: Authenticated Initial Key Agreement

Join/Merge - As before each joining participant, $U_i (i \in \mathcal{J})$, sends a $JOIN$ request along with its identity, U_i, random nonce, N_i and blinded secret, g^{r_i}. The old group leader (U_l) chooses a new random secret, r_l, and sends all the blinded secrets to the new group leader, $U_{l'}$, (which can be chosen randomly). The new group leader broadcasts a message similar to the round 3 message in **IKA** *i.e.* all the blinded secrets and the blinded secrets raised to his (new) secret. See Table 3.8 for formal specification.

Delete/Partition - When participants leave the group, the group leader changes his secret contribution and sends an **IKA** Round 3 like message to the group, omitting the leaving participants' contributions. Refer to Table 3.9 for details.

3.2.4 Security Analysis

The Security model is as before and thus we directly present the security result for this authenticated version of the protocol.

Theorem 3.4 *Let P be the protocol as defined in the last section. Let $\mathcal{A}$ be an active*

Round 1
$\forall i \in \mathcal{J}, r_i \stackrel{r}{\leftarrow} [1, q-1], N_i \stackrel{r}{\leftarrow} \{0,1\}^k,$
$U_i \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_i = \{JOIN, U_i, N_i, g^{r_i}\}, \sigma_i\}$
Round 2
$\forall i \in \mathcal{J}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)\ r_l \stackrel{r}{\leftarrow} [1, q-1], l' \stackrel{r}{\leftarrow} \mathcal{M} \cup \mathcal{J}$
$U_l \longrightarrow U_{l'} : \{msg_l = \{JREPLY, \{U_i, N_i, g^{r_i}\}_{\forall i \in \mathcal{M} \cup \mathcal{J}}\}, \sigma_l\}$
Round 3
$if(\mathcal{V}_{PK_i}\{msg_l, \sigma_l\} == 1), l \leftarrow l', r_l \stackrel{r}{\leftarrow} [1, q-1], \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{J}$
$U_l \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_l^2 = \{JGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l^2\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} and N_i are as contributed
$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 3.8: Authenticated Join/Merge protocol

Round 1
$\forall i \in \mathcal{D}, U_i \longrightarrow U_l : \{msg_i = \{DEL, U_i, N_i\}, \sigma_i\}$
Round 2
$\forall i \in \mathcal{D}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1), r_l \stackrel{r}{\leftarrow} [1, q-1], \mathcal{M} \leftarrow \mathcal{M} \setminus \mathcal{D}$
$U_l \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_l = \{DGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l, \sigma_l\} == 1)$ and g^{r_i} and N_i are as contributed
$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 3.9: Authenticated Delete/Partition protocol

adversary making q_{ex} ***Execute*** *queries and q_s* ***Send*** *queries to the participants and running in time t. Then Protocol P is a secure GKA protocol. Namely:*

$$\mathrm{Adv}_P(t, q_{ex}, q_s) \leq \mathrm{Adv}_{DDH}(t') + |\mathcal{P}| * Succ_{\Sigma}(t') + \frac{q_s^2 + q_{ex}q_s}{2^k} + \frac{1}{|G|}$$

where $t' \leq t + |\mathcal{P}|(q_{ex} + q_s)(t_{exp} + t_{lkup})$, t_{exp} is the time to perform an exponentiation in G, k is a security parameter and t_{lkup} is the time to perform a look-up in tables L and $Sessions$, to be defined in the proof.

Proof: Let $\mathcal{A}$ be an adversary that plays in the game G_0 against the protocol P. We will define a series of games $G_1, \ldots G_5$ such that each game G_i differs "slightly" from its precedent game G_{i-1}. We denote the event Win in the game G_i by Win_i. Thus by explicitly quantifying the effect the slight difference in the games has on the winning probability of the adversary, one can relate the winning probability of $\mathcal{A}$ in the original game G_0 ($\Pr[Win_0]$) to any other game. We stop when we eventually reduce to a simple game (here G_5) for which we can calculate $\Pr[Win_i]$. Thus by relating all the probabilities we can eventually calculate $\Pr[Win_0]$.

All queries made by $\mathcal{A}$ are answered by a challenger Δ. Δ maintains two tables: *Sessions* and L. In table *Sessions*, Δ keeps transcripts of each and every session generated by him (either with a single **Execute** query or multiple **Send** queries). While in table L, Δ maintains a list of all *blinded secrets* generated by him during the game and their corresponding secrets.

Game G_0: This game G_0 is the real game as defined earlier. Δ initializes the game by generating public-private key pairs for all the participants as specified by the protocol and choosing a random bit b, which is used by him to answer the **Test** query. Then it answers all queries of the adversary in accordance with the protocol P.

Game G_1: The game G_1 is identical to G_0 except that Δ aborts if a signature forgery occurs for some player U before any **Corrupt(U)** query was made. We denote such an event by E_1. Using a well-know lemma [Sho] we get: $|\Pr[Win_0] - \Pr[Win_1]| \leq \Pr[E_1]$. Note that Δ can detect a signature forgery for some player U when he detects

a valid message, not generated by him (all messages generated by Δ are stored in the *Sessions* table), in some session before the **Corrupt** query was made to U.

Calculation of $\Pr[E_1]$: The event E_1 occurs when the adversary makes an existential signature forgery for any one of the protocol participants. The probability of this happening is bounded by $|\mathcal{P}| * Succ_{\Sigma}(t')$ where $Succ_{\Sigma}(t')$ is the maximum success probability of an existential signature forgery against a signature scheme Σ running in time t', given some public key PK.

Game G_2: The game G_2 is identical to G_1 except that Δ aborts if a nonce used in some **Send** query has already been used in some **Execute** or **Send** query before. We denote the occurrence of the nonce being repeated in some **Send** query as event E_2. Then: $|\Pr[Win_1]$ - $\Pr[Win_2]| \leq \Pr[E_2]$. Δ can detect event E_2 as he can track all nonces generated, via the *Sessions* table.

Calculation of $\Pr[E_2]$: Clearly the probability of event E_2 happening is $\frac{q_s(q_{ex}+q_s)}{2^k}$.

Game G_3: In game G_3, Δ modifies the way it answers the queries slightly. Δ is given as input a DDH-tuple (g, g^{r_a}, g^{r_b}, $g^{r_a r_b}$). Δ follows the protocol as before to generate query responses but changes the way it generates the *blinded secrets* used in the transcript. The change is as follows:

Whenever a *blinded secret* is to be generated for some session participant M_i, instead of raising the group generator g to a randomly chosen number r_i (from $[1, q-1]$, as specified by the protocol), it raises g^{r_b} (from the given tuple) to α_i and g to β_i and uses the value $g^{r_b \alpha_i} g^{\beta_i}$ as the *blinded secret* for participant M_i in the transcript. Both α_i and β_i are randomly chosen from $[1, q-1]$. The corresponding *blinded response* is generated as before by raising the *blinded secret* to the group leader's secret r_l (which is also randomly chosen from $[1, q-1]$, as specified by the protocol). Also, Δ stores the *blinded secret* so generated and (α_i, β_i) in table L. Table 3.10 illustrates the difference between *blinded secrets* between Game G_2 and Game G_3. In this way, Δ knows all *blinded secrets* generated by him during the game and their corresponding *secrets*. Clearly from the adversary's point of view there is no change in the game. Thus $|\Pr[Win_2] = \Pr[Win_3]|$.

	Game G_2	Game G_3
Blinded Secret (bs_i)	g^{r_i}	$g^{(r_b\alpha_i+\beta_i)}$
Blinded Response (br_i)	$g^{r_i r_l}$	$g^{(r_b\alpha_i+\beta_i)r_l}$
Key (k_i)	$g^{r_l} * \Pi_i br_i$	$g^{r_l} * \Pi_i br_i$

Table 3.10: Difference in Query Responses between Game G_2 and Game G_3

Game G_4: Game G_4 is same as game G_3 except that Δ modifies the way it generates the *blinded responses*. Using the same DDH-tuple (g, g^{r_a}, g^{r_b}, $g^{r_a r_b}$), Δ does the following:

Whenever a *blinded response* is to be generated for some session participant M_i, Δ retrieves all the *blinded secrets* for that session from the table $Sessions$. Then it looks for these *blinded secrets* in table L. If Δ finds all these *blinded secrets* in the table, it retrieves the corresponding *secret* entry (α_i, β_i) for M_i and chooses randomly α_l from $[1, q-1]$. It uses the value $(g^{r_a r_b})^{\alpha_i}(g^{r_a})^{\beta_i}(g^{r_b})^{\alpha_i\alpha_l} g^{\beta_i\alpha_l} = g^{(r_b\alpha_i+\beta_i)(r_a+\alpha_l)}$ as the *blinded response* for participant M_i in the session transcript. If on the other hand, Δ does not find some *blinded secret* of that session in table L, this means that this *blinded secret* has been introduced by the adversary $\mathcal{A}$ and Δ does not know the corresponding secret. Thus for a session where any of the *blinded secrets* is not found in table L, Δ continues to generate the *blinded responses* (for all the participants) as in game G_3 (by raising *blinded secret* to the *secret* of the leader).

Thus, in brief, Δ uses the value $g^{(r_b\alpha_i+\beta_i)(r_a+\alpha_l)}$ as the *blinded response* for participant M_i, if all *blinded secrets* in that session were generated by him; otherwise it uses the value $g^{(r_b\alpha_i+\beta_i)r_l}$ (see Table 3.11). Note that as $\mathcal{A}$ can make a **Test** query only on a *fresh* participant instance, this rules out those sessions where $\mathcal{A}$ has been able to introduce *blinded secrets* on his own (by making **Corrupt** and then **Send** queries). Thus Δ can respond to the queries of such sessions without using data from the DDH-tuple[4].

[4]As Δ does not have to guess the query for which he wants to use data from the DDH-tuple, there is no q_s or q_{ex} factor in the final security reduction.

	Game G_3	Game G_4
Blinded Secret (bs_i)	$g^{(r_b\alpha_i+\beta_i)}$	$g^{(r_b\alpha_i+\beta_i)}$
Blinded Response (br_i)	$g^{(r_b\alpha_i+\beta_i)r_l}$	$g^{(r_b\alpha_i+\beta_i)(r_a+\alpha_l)}$
Key (k_i)	$g^{r_l} * \Pi_i br_i$	$g^{(r_a+\alpha_l)} * \Pi_i br_i$

Table 3.11: Difference in Query Responses (for possible **Test** sessions) between Game G_3 and Game G_4

Clearly again from the adversary's point of view there is no change in the game. Thus $|\Pr[Win_3] = \Pr[Win_4]|$.

Game G_5: Game G_5 is same as Game G_4 except that instead of a DDH-tuple (g, g^{r_a}, g^{r_b}, $g^{r_a r_b}$), Δ's input is a random tuple (g, g^{r_a}, g^{r_b}, g^{r_c}), where all r_a, r_b and r_c are randomly drawn from [1, q] and $r_c \neq r_a r_b$. Δ continues answering the queries as in Game G_4, except that the role of $g^{r_a r_b}$ is taken by g^{r_c}. And now when answering the **Reveal** query or **Test** query (in case bit $b = 1$), Δ uses g^{r_a} in computing the session key instead of $g^{\frac{r_c}{r_b}}$ which the protocol demands.

Thus, in brief, Δ uses the value $(g^{r_c})^{\alpha_i}(g^{r_a})^{\beta_i}(g^{r_b})^{\alpha_i\alpha_l}g^{\beta_i\alpha_l} = g^{(r_c - r_a r_b)\alpha_i + (r_b\alpha_i+\beta_i)(r_a+\alpha_l)}$ as the *blinded response* for participant M_i, if all *blinded secrets* in that session were generated by him otherwise it uses the value $g^{(r_b\alpha_i+\beta_i)r_l}$.

Claim 1: $|\Pr[Win_4]$ - $\Pr[Win_5]| \leq \text{Adv}_{DDH}(t') + \frac{1}{|G|}$, where t' is bounded by $t + |\mathcal{P}|(q_{ex} + q_s)(t_{exp} + t_{lkup})$, t_{exp} being the time to perform an exponentiation in G and t_{lkup} the time to perform a look-up in tables L and $Sessions$.

Proof: Consider a hybrid game G' which interpolates between G_4 and G_5 depending on its input. Game G' is same as game G_4 (and game G_5) except that the input to G' is a DDH challenge (g, g_1, g_2, g_3). Thus when the DDH challenge is a DDH tuple, the game proceeds exactly like game G_4 while when the DDH challenge is a non-DDH tuple it proceeds exactly as game G_5, that is:
$|\Pr[r_a, r_b \overset{r}{\leftarrow} \mathbb{Z}_q : D(g, g^{r_a}, g^{r_b}, g^{r_a r_b}) = 1] - \Pr[r_a, r_b \overset{r}{\leftarrow} \mathbb{Z}_q, r_c \overset{r}{\leftarrow} \mathbb{Z}_q \setminus \{r_a r_b\} : D(g, g^{r_a}, g^{r_b}, g^{r_c}) = 1]| \geq |\Pr[Win_4]$ - $\Pr[Win_5]|$

or

	Game G_4	Game G_5
Blinded Secret (bs_i)	$g^{(r_b\alpha_i+\beta_i)}$	$g^{(r_b\alpha_i+\beta_i)}$
Blinded Response (br_i)	$g^{(r_b\alpha_i+\beta_i)(r_a+\alpha_l)}$	$g^{e\alpha_i+(r_b\alpha_i+\beta_i)(r_a+\alpha_l)}$
Key (k_i)	$g^{(r_a+\alpha_l)} * \Pi_i br_i$	$g^{(r_a+\alpha_l)} * \Pi_i br_i$

Table 3.12: Difference in Query Responses (for possible **Test** sessions) between Game G_4 and Game G_5, where $e = r_c - r_a r_b$

r_a, r_b, r_c, $\{\alpha_i\}$, $\{\beta_i\} \stackrel{r}{\leftarrow} [1, q-1]$ $(r_c \neq r_a r_b)$
$x_1 = g^{(r_b\alpha_1+\beta_1)}, \ldots, x_n = g^{(r_b\alpha_n+\beta_n)}$
$y_1 = g^{e\alpha_1+(r_b\alpha_1+\beta_1)(r_a+\alpha_l)}, \ldots, y_n = g^{e\alpha_n+(r_b\alpha_n+\beta_n)(r_a+\alpha_l)}$
$z = g^{(r_a+\alpha_l)} * \Pi_{i=1}^{n} y_i$
$T = (x_1, \ldots, x_n, y_1, \ldots, y_n, z)$

Table 3.13: Session Transcript, T, with n participants

$|\Pr[r_a, r_b \stackrel{r}{\leftarrow} \mathbb{Z}_q : D(g, g^{r_a}, g^{r_b}, g^{r_a r_b}) = 1] - \Pr[r_a, r_b, r_c \stackrel{r}{\leftarrow} \mathbb{Z}_q : D(g, g^{r_a}, g^{r_b}, g^{r_c}) = 1]|$ $+ \frac{1}{|G|} \geq |\Pr[Win_4]$ - $\Pr[Win_5]|$.

Thus $\mathrm{Adv}_{DDH}(t') \geq |\Pr[Win_4]$ - $\Pr[Win_5]|$ - $\frac{1}{|G|}$.

Claim 2: $\Pr[Win_5] = \frac{1}{2}$.

Proof: It suffices to show that the session key of the **Test** session is uniformly distributed in G independent of anything else. The distribution of a (possible **Test**) session transcript (T) with $n+1$ participants (including the group leader) is shown in Table 3.13, where x_i's represent the *blinded secrets*, y_i's represent the *blinded responses* and z the session key (which the adversary acquires by making the **Reveal** query). We have removed the nonces, identifiers and signatures in this transcript. Please note that the maximum number of participants in a session can be p, the order of the participant set $\mathcal{P}$. Across sessions, r_a, r_b and r_c are fixed while α_i's, β_i's and α_l are different (some α_i's and β_i's can be same across the sessions) We will show that z in a possible **Test** session is independent of the values x_i's, y_i's in that session and x_i's, y_i's and z's in other sessions. In fact it suffices to show that z in some fixed session is independent of the values x_i's, y_i's in that session, because z is a function of α_l and

y_i's, which are ensured to be different (by a different choice of α_l) in each session (see Table 3.13). There can be some common x_i's between two sessions but that does not affect the distribution of the session key.

Therefore we want to show that,
$\Pr[z = c | x_1 = a_1, x_2 = a_2, \ldots, x_{p-1} = a_{p-1}, y_1 = b_1, y_2 = b_2, \ldots, y_{p-1} = b_{p-1}] = \frac{1}{q}$, where without loss of generality we take the session size to be p.

The following relation holds between the quantities x_i's, y_i's and z (in the following $\log x$ denotes $\log_g x$).

$\log x_1 = r_b\alpha_1 + \beta_1$
$\log x_2 = r_b\alpha_2 + \beta_2$
$\ldots \quad \ldots$
$\log x_{p-1} = r_b\alpha_{p-1} + \beta_{p-1}$
$\log y_1$ - $\log x_1(\log z$- $\Sigma_{i=1}^{p-1} \log y_i) = e\alpha_1$
$\log y_2$ - $\log x_2(\log z$- $\Sigma_{i=1}^{p-1} \log y_i) = e\alpha_2$
$\ldots \quad \ldots$
$\log y_{p-1}$ - $\log x_{p-1}(\log z$- $\Sigma_{i=1}^{p-1} \log y_i) = e\alpha_{p-1}$
$\log z = r_a + \alpha_l + \Sigma_{i=1}^{p-1} \log y_i$

We fix the quantities x_i's and y_i's to their respective values (a_i's and b_i's) and then fix z to some random value c. If for each choice of c the above system of equations has a unique solution (for the unknowns α_i's, β_i's and α_l), we have the desired result. The above system can be re-written as follows:

$A_1 = r_b\alpha_1 + \beta_1$
$A_2 = r_b\alpha_2 + \beta_2$
$\ldots \quad \ldots$
$A_{p-1} = r_b\alpha_{p-1} + \beta_{p-1}$
$B_1 = e\alpha_1$
$B_2 = e\alpha_2$
$\ldots \quad \ldots$
$B_{p-1} = e\alpha_{p-1}$

$C = \alpha_l$

where $C = \log c$ - r_a - $\Sigma_{i=1}^{p-1} \log b_i$, $A_i = \log a_i$ and $B_i = \log b_i - \log a_i(\log c - \Sigma_{i=1}^{p-1} \log b_i)$.

This system of equations can be re-written in matrix form as follows:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & & & \cdots & 0 \\ 0 & 1 & 0 & \cdots & r_b & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & \cdots & r_b & 0 & \cdots & 0 \\ \vdots & & & \ddots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & \cdots & & r_b \\ 0 & 0 & 0 & \cdots & 0 & e & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & e & \cdots & 0 \\ \vdots & & & & \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & e \end{pmatrix} \cdot \begin{pmatrix} \alpha_l \\ \beta_1 \\ . \\ . \\ \beta_{p-1} \\ \alpha_1 \\ . \\ . \\ \alpha_{p-1} \end{pmatrix} = \begin{pmatrix} C \\ A_1 \\ . \\ . \\ A_{p-1} \\ B_1 \\ . \\ . \\ B_{p-1} \end{pmatrix}$$

The determinant of the coefficient matrix is e^{p-1} which is not equal to 0 as $e = r_c - r_a r_b$ is assumed to be non-zero. That implies this system of equation has a unique solution. That is to say even if the quantities x_i's and y_i's are fixed, z (the session key) is uniformly distributed. Hence the adversary can have no advantage in distinguishing the session key in Game G_5. That is $\Pr[Win_5] = \frac{1}{2}$.

Combining all the above results, we get: $\Pr[Win_0] \leq \Pr[E_1] + \Pr[E_2] + \mathrm{Adv}_{DDH}(t') + \frac{1}{|G|} + \frac{1}{2}$ and so the desired result follows.

3.2.5 Implementation

In this section, we present our results of implementation of our unauthenticated version of the protocol in the context of a single-hop ad hoc network and also study how the authenticated protocol may be employed in a multi-hop ad hoc network.

To test the performance of the unauthenticated GKA protocol, we incorporated it

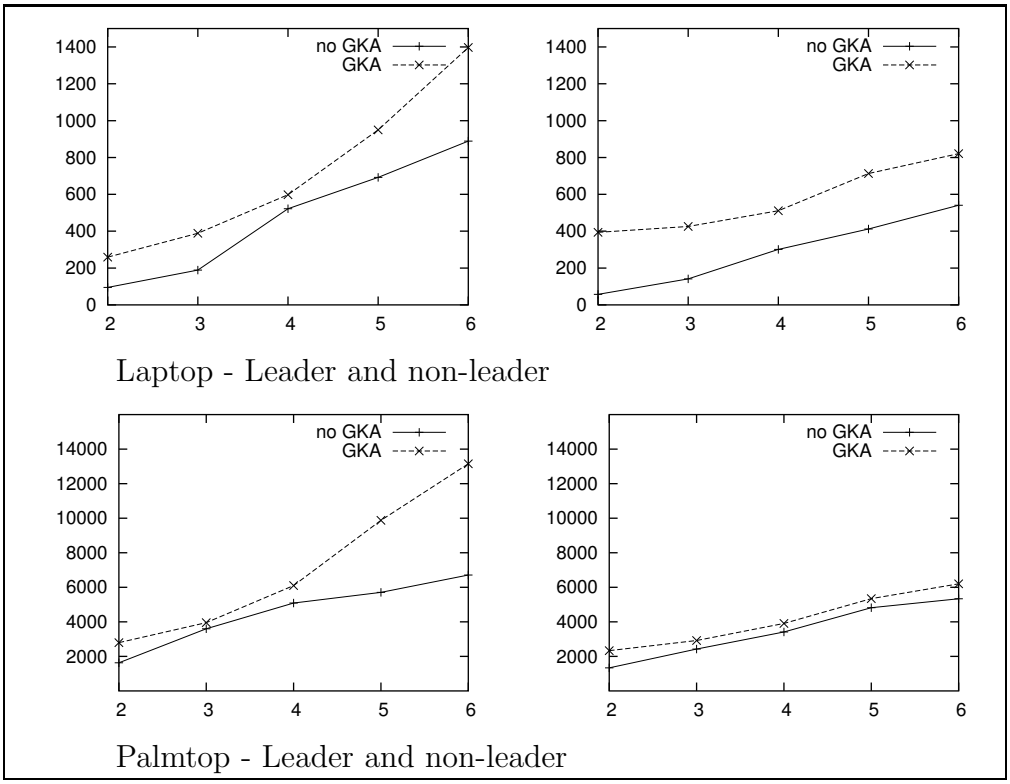


Figure 3.2: Computation time (in msec) per device with and without GKA

in the group management protocol of AdhocFS [BI03]. The group management of [BI03] consists of three communication rounds: $DISC$, $JOIN$ and $GROUP$. The $DISC$ stage initiates the group formation by calling for interested participants. Each interested participant responds with a $JOIN$ message. The group membership is defined and announced by the group leader (chosen randomly) by the $GROUP$ message. The design of the new GKA protocol allowed us to piggy-back GKA data on group management messages, thus member contributions towards the group key are collected during $JOIN$ messages while the $GROUP$ message carries the message from the group leader which enables everyone to compute the group key. Thus no additional communication round is required to derive a group key, irrespective of the group size. It is worth mentioning that it would not have been possible with most of the protocols presented in Table 3.1, as the messages sent by group members are dependent on messages sent by other members. The computation times on a device in the absence and presence of GKA procedures are plotted in Figure 3.2. The data shown is for an experimental setup consisting of laptops (Compaq 500 Mhz running Linux) and palmtops (Compaq ipaq 400MHz running Linux familiar 0.7). All random contributions for the group key were chosen from a Diffie-Hellman group of prime order of 1024 bits. The code was written in Java except the exponentiation function which was implemented in native code with the GMP library [Lib]. The graphs in Figure 3.2 plot computation time (in milliseconds on Y axis) against group-size with and without GKA. There are separate plots for the cases when the device was a leader/non-leader. As expected, the time for non-leader members increases (when employing GKA protocol) by an almost constant factor (order of time to perform two 1024 bit exponentiations), while for a leader it increases linearly as the group size increases. As most ad hoc networks are expected to be composed of devices of unequal computing power, more powerful devices (like laptops) can assume the role of a leader more often. Thus our GKA protocol performs well in a single-hop ad hoc network with the first node to issue the $DISC$ request being selected as the group leader.

We now present implementation details of our authenticated protocol in a multi-hop network. For that, we consider medium sized (15 - 50 nodes) multi-hop ad hoc

networks with some routing protocol providing connectivity to the network. We also assume the presence of a broadcast mechanism to flood messages within the network. For larger networks, deriving a group key from the contribution of each node can be quite time consuming unless only certain nodes are selected to participate in the protocol.

Election of a group leader - The GKA protocol that we propose is asymmetric in the role of the members in the group. One unique member in each session initiates the protocol and does more computational work than others. Though we call him the group leader, he does not have any special authority or "cryptographic ability" which makes him different from others. In fact the advantage in making use of a group leader is that it makes it easier to define the composition of a group with a minimum number of messages. For instance, our protocol can be implemented without the need for any group leader. Anyone (and everyone) could issue an $INIT$ request and interested participants could respond to it. In this way we would have a large number of messages being flooded in the network and often end up with multiple groups with overlapping membership. But if only one node in the network issued the $INIT$ request, one could arrive at defining one single group (out of the connected component of the network) with much less traffic. This particular node might be changed (for various reasons) in the next session when the keys needs to be changed. But of course the question, then is, how to choose the node which issues the $INIT$ request ? In some applications, the answer could come from the logistics of the application itself. For example, as already observed, our GKA protocol integrated well with the AdhocFS [BI03] application where there was already a need for a group leader to check the connectivity of the network and the membership of the group. But for applications which have no inbuilt need for a group leader, the situation is a bit more complicated. Leader election protocols for asynchronous and lossy networks, which ensure that exactly one leader is elected at the end of the protocol (and all other nodes know the leader) tend to be highly complex. A lot of literature exists in this area and includes [ORV94, Sin97]. The best protocol requires $n \log_2 n$ rounds to decide a unique leader. Thus the use of such protocols will be way too expensive in terms of the number of messages in most scenarios. Randomized or Probabilistic

leader election protocols [GvRB00, MWV00] are more efficient (though can be still complex) but only provide certain probabilistic guarantees about the uniqueness of the leader. The latter category of protocols are more suited for MANET scenarios. We present a much simpler approach to resolve the issue of group leader election in our protocol. The idea is to allow any member to issue an $INIT$ request if it detects an absence of the group leader. In the case of multiple leaders being present in the network, the number of leaders are reduced slowly to one using simple rules. Thus the protocol converges to a single leader but possibly not in a single session but over a period of few sessions. During this convergence period, the protocol continues to handle the group dynamics. Convergence of multiple group leaders and handling of group changes can be efficiently done by requiring both the current group leader and other participants to periodically broadcast their group key contributions. Thus we retain the following messages from our protocol:

- $IGROUP$: This message can be broadcast by any node (with its ID and nonce), when it fails to detect a group leader. It is also broadcast by the current group leader with the current group composition and the *blinded secrets* and *blinded responses*. For a new incoming node, this message serves as a welcome message whereby it can know the existing group composition and its leader. The new node can join this group by sending a $IREPLY$ message with its contribution. For other members in the group, the $IGROUP$ message serves as a check of their view of the group composition and the group leader. They can also check if their contribution is well included in the group key. During the period of its leadership, the group leader is required to re-broadcast the $IGROUP$ message with some fixed period T (details later). This message could remain unchanged if there are no group changes, and could reflect the change (if any) by adding the contributions and IDs of new members and deleting those of leaving members.

- $IREPLY$: This message is meant for members to join the group or re-affirm their presence in the group. Thus new members can reply with their contribution while old members can simply re-send this message to affirm their presence

in the group. They can also change their group key contribution via this message. A group member is required to periodically broadcast this message with period T.

- DEL: This message can by sent by any node to leave the group. As we explain later, lack of consecutive $IREPLY$ messages from any node is also considered as a leave by that node, we retain this message as a fast mechanism to detect leaving members.

Thus with these messages, group formation and group key calculation take place as follow: On arriving in a network, a node listens for a $IGROUP$ message for a fixed period $k * T$ (where both k and T are known a priori by all nodes). Since $IGROUP$ messages can be lost, k must be selected so that the probability that $k-1$ successive transmissions of $IGROUP$ message is lost is small. If it does not receive any such message by the end of this period, it sets a random timer in the interval $[rtd, l * rtd]$, where rtd is the round trip delay time of the network and l is an integer. If by the expiry of this timer, no $IGROUP$ message is received, the node broadcasts its own $IGROUP$ message including its ID and nonce. The idea of choosing this interval is to avoid two participants from sending $IGROUP$ messages at the same time, after having detected lack of a group leader. Thus, for example, a node choosing the timer period as $7 * rtd$ is likely to receive the $IGROUP$ message from a node who chose it as $5 * rtd$, before the former's timer expires. To further reduce the likelihood of a collision, a node may add a small jitter (time delay) before sending its $IGROUP$ message. l must be a function of the total number of nodes in the network and such that the probability of two nodes choosing the same number from [1, l] is low. On the other hand if a node receives one or more $IGROUP$ message before the expiry of this random timer, it should abort the process of sending an $IGROUP$ message. Instead it should choose one $IGROUP$ message to which it can reply with a $IREPLY$ message. A simple (and same for all) decision rule (for instance group leader with lower ID or group with more members) could be used to decide which $IGROUP$ message to reply to. After having chosen the group, the node sends a $IREPLY$ message to the group leader with its contributions.

On receiving these $IREPLY$ messages, the group leader starts updating the $IGROUP$ message that it will send at the next expiry of period T. When this new $IGROUP$ message is broadcast, older members can be updated with the new view of the group, and calculate the new group key (using their old secret). The new members can see that their $IREPLY$ message was well-received and can calculate the group key. All group members must periodically broadcast their $IREPLY$ message with period T (and a small jitter). This ensures the group leader of the presence of the members. The contributions of the members and the group leader towards the group secret must be changed often to maintain key independence and secrecy. The property of *key independence* requires that the group key be changed at every group membership change and to achieve that it suffices that the current group leader change its contributed secret. But in highly dynamic and large networks, it may be quite an expensive operation to compute a new group key every time the group membership changes. Thus additional rules to change the group key could be used, depending on the dynamics of the network. For instance, the key could be changed if some fixed number of members leave or when some fixed period P expires.

The factors affecting the choice of these parameters are detailed in Table 3.14. The failure of the current group leader should automatically cause an auto-election. Re-election of the group leader can also be initiated at the expiry of the fixed period P (same as that for changing the key). Tables 3.15 and 3.16 detail the procedures followed by a leader node and a participant node respectively.

Handling merge and partitions - The periodic broadcast of $IGROUP$ messages makes the task of handling mergers and partitions in the network straightforward. Thus when two or more groups with their respective leaders merge, the same simple rule of leader election (group size or lower ID of the leader) helps everyone decide which leader to continue with. Thus nodes (including the current group leader) on receiving multiple $IGROUP$ messages, decide which node will be the new leader. In case of a change of leadership, each node stops sending $IREPLY$ messages (or send an explicit DEL message) to the old group leader and unicasts its new group key contribution to the new leader. If the group leader does not receive a $IREPLY$

Parameter	Constraint
Time-period for change of key/leader, P	small enough to take group dynamics into account
Time-period for re-broadcast of $IGROUP$ message, T	small enough to take dynamics of the group into account
Number of $IGROUP$ time-periods to wait before sending an $INIT$, k	large enough to be sure the message is not simply lost
Length of the back-off interval (in rtd), l	large enough to avoid collision during the group leader election
Round trip time delay of the network, rtd	-

Table 3.14: Protocol parameters

```
Leader (k, T, l, rtd) {
    set-timer( t1, T, timer-expiry-handle-fn1() );
    while (1) {
        listen(msg);
        if (msg.type == IGROUP)
            if (msg.senderID ≤ myID) exec(Participant(k, T, l, rtd, msg.senderID)) ;
            endif;
        else if (msg.type == IREPLY or DEL) updateIGROUP();
        endif;
    }
timer-expiry-handle-fn1() {
    reset-timer(t1);
    send-msg(IGROUP);
}
}
```

Table 3.15: Pseudo-code for leader

message from a particular node for a given number of periods, the lack of reception of these messages should be handled as the reception of a DEL message. In such a case appropriate action must be taken by the group leader which should recompute another $IGROUP$ message.

3.3 Key Control and Key Confirmation

Traditionally group key agreement protocols have not considered the case when some of the participants (or collusion of participants) of the protocol have malicious intentions. The subject has been sparingly dealt for the case of two parties in [MWW98] and multiple parties in [PW04]. In ad hoc networks, where some of the nodes could be possibly compromised, the issue does indeed look relevant.

The work of Wang *et al.* [PW04] identifies the "pre-fixing" of the key by some malicious participant as a possible attack. The fact that any of the participants in a group key agreement protocol cannot pre-determine the key before the actual execution of the protocol, makes it different from a key transport protocol. Although the group leader in our protocol chooses its contribution after other members have chosen theirs, it does not imply the group leader can pre-determine the group key. In fact like many other protocols (including GDH.2, GDH.3, TGDH, NKYW in Table 3.1), the group member choosing last his contribution might have some advantage but it does not translate to key control as discussed in [AG00, PW04]. We show below that the group leader in our protocol cannot fix the group key to a given value K_f. The group key $g^{r_l(1+\sum_{i\in\mathcal{M}\setminus\{l\}} r_i)}$ can be viewed as a two-party Diffie-Hellman key where one participant's contribution is g^{r_l} and the other's $g^{(1+\sum_{i\in\mathcal{M}\setminus\{l\}} r_i)}$. Denoting $g^{(1+\sum_{i\in\mathcal{M}\setminus\{l\}} r_i)}$ by g^B, the group leader needs a polynomial time algorithm $\mathcal{A}$ which given the desired group key $K_f = g^{r_K}$ and g^B as inputs can output $g^{\frac{r_K}{B}}$ to be used as g^{r_l} i.e. $\mathcal{A}(g^{r_K}, g^B) = g^{\frac{r_K}{B}}$. But in fact this algorithm can be used to solve the computational Diffie-Hellman problem as follows: Given g^α and g^β, $\mathcal{A}(g^\alpha, g^\beta) = g^{\frac{\alpha}{\beta}}$; $\mathcal{A}(g^{\frac{\alpha}{\beta}}, g^\alpha) = g^{\frac{1}{\beta}}$; $\mathcal{A}(g^\alpha, g^{\frac{1}{\beta}}) = g^{\alpha\beta}$.

```
Participant(k, T, l, rtd, current-leader-ID) {
    if (current-leader-ID == infinity) {
       set-timer( t_1, k * T, timer-expiry-handle-fn1() );
       set-timer( t_3, T, timer-expiry-handle-fn3() );
       }
    else prepare-mesg(IREPLY, current-leader-ID);
    while (1) {
       listen(msg);
       if (msg.type == IGROUP) process-IGROUPmsg(msg);
       else if (msg.type == IREPLY) exec(Leader(k, T, l, rtd));
       endif;
    }

process-IGROUPmsg(msg) {
    reset-timer(t_1);
    if (msg.senderID ≤ current-eader-ID) {
       current-leader-ID = msg.senderID;
       prepare-mesg(IREPLY, current-leader-ID);
    endif;
}

timer-expiry-handle-fn1() {
    set-timer( t_2, l * rtd, timer-expiry-handle-fn2() );
}

timer-expiry-handle-fn2() {
    reset-timer( t_2);
    reset-timer( t_1);
    send-msg( IGROUP);
}

timer-expiry-handle-fn3() {
    if (current-leader-ID != infinity) send-msg(IREPLY);
}
}
```

Table 3.16: Pseudo-code for participant

Other kinds of attacks that a malicious insider (or collusion of insiders) could mount are not well studied. One general attack, wherein a malicious insider succeeds in sharing different keys with different members in the group, while they all think they have the same shared key is discussed in [TM04]. A simple but expensive technique to resist such attacks is to add a round of key confirmation to the key agreement protocol. In the key confirmation round, each participant is required to simultaneously broadcast a well-known quantity encrypted with the computed session key.

The security model present in this dissertation does not address this issue and no GKA protocol (known to us) has been proved secure against such attacks (without the use of a key confirmation round). Very recent work of Katz and Shin [KS05] is the first attempt to formally model attacks by malicious insiders. Thus it is of interest in the future to provide security proofs in this new evolving model.

But we can further prevent any control of the key by the group leader (in our protocol) by requiring it to commit to its contribution before others as follows: The current group leader sends also the hash value of his contribution in the $INIT$ message ($H(g^{r_l})$ in Table 3.17). Thus the group leader commits to the hash of its contribution before it has received others' contribution. All participants then confirm that the hash of the group leader's contribution (which is revealed in the $IGROUP$ message) is indeed same as that in the $INIT$ message. We believe the security proof of this protocol can be similarly constructed in the *Random Oracle Model* [BR93].

3.4 Conclusion

In this chapter we presented unauthenticated and authenticated versions of a new group key agreement protocol for ad hoc networks. The protocol is particularly suited to ad hoc networks, but it compares well to existing protocols proposed for Internet group communication. The protocol requires no well-defined structure in the network, nor any ordering among the participants. Any node can assume the role of the group leader if its absence is detected and initiate the GKA protocol. The protocol is robust

Round 1
$l \stackrel{r}{\leftarrow} \mathcal{M}, N_l \stackrel{r}{\leftarrow} \{0,1\}^k$, $r_l \stackrel{r}{\leftarrow} [1, q-1]$
$U_l \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_l^1 = \{INIT, U_l, N_l, H(g^{r_l}), \sigma_l^1\}$
Round 2
$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_l}\{msg_l^1, \sigma_l\} == 1)$, $r_i \stackrel{r}{\leftarrow} [1, q-1], N_i \stackrel{r}{\leftarrow} \{0,1\}^k$,
$U_i \longrightarrow U_l : \{msg_i = \{IREPLY, U_l, N_l, U_i, N_i, g^{r_i}\}, \sigma_i\}$
Round 3
$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)$
$U_l \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_l^2 = \{IGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l^2\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} and N_i are as contributed and hash value of g^{r_l} matches that sent in **Round 1**
$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 3.17: Modified IKA

to loss of messages, as it not "halted" in case of loss of a few messages. We proved the security of these protocols in existing models of security, and obtained a tight reduction. We discussed favourable results of our implementation in experimental set-ups. In the next chapter we study the area of secure routing in ad hoc networks.

Chapter 4

Secure Routing in Ad hoc Networks

As discussed in Chapter 2, routing functions in mobile ad hoc networks are carried out by all available nodes in the network. This is quite different from traditional networks where only dedicated nodes participate in the routing process. While this participation by the whole network in the routing process makes it more efficient, it makes it harder to secure. This is not only because now the number of nodes available for mounting attacks on routing is much higher, but also because, all of them cannot be physically secured from possible adversaries. A presence of a single malicious node in the network is all that may be required to disrupt the routing process. As discussed in Section 2.3.3, the motive of an adversary can be varied: a) forcing a certain route to go via it, b) forcing a certain route to not go via a certain node or c) simply deteriorating the performance of the routing. And to achieve them, the adversary may mount both *passive* and *active* attacks. Also the adversary may be an *insider node* or an *outsider node*. By an *insider node*, we refer to a node which is known by other nodes to be part of the network and eligible to participate in the routing process. While an *outsider node* is a node whose presence is unknown (and unacceptable) to other nodes in the network. The goal of an insider node may be to gain some advantages without being caught in the act, while that of an outsider node

could be to simply remain hidden and continue mounting the attacks. The mobility of nodes and lossy nature of the network makes the task of detecting *outsider nodes* quite difficult. In the following sections, we discuss the prior work done in secure routing of ad hoc networks and then introduce a new cryptographic primitive, an Aggregate Designated Verifier Signature, and show how it is used to secure routing in Ad hoc networks.

4.1 Background and Existing work

Both cryptographic and non-cryptographic mechanisms have been proposed to achieve secure routing in ad hoc networks. Non-cryptographic mechanisms include use of location information of a node [CY02], and use of multiple routes, between two nodes, to achieve robustness [PH03a] as well as reputation and incentive based mechanisms. We only discuss work proposing cryptographic mechanisms to secure routing in this thesis. Routing protocols for ad hoc networks can be classified into three categories, namely, a) Pro-active routing protocols, b) Reactive routing protocols and c) Hybrid routing protocols. Examples of secure pro-active routing protocols include SEAD [HPJ02b] and SLSP [PH03b], while examples of secure reactive routing protocols are SRP [PH02], ARIADNE [HPJ02a], ARAN [SDL+03], SAODV [Zap] and endairA [BV04]. ZRP [zrp] is an example of a secure hybrid routing protocol. While pro-active routing protocols provide better performance in ad hoc networks with low latency requirements, reactive routing protocols have been shown to perform better in most ad hoc network scenarios [MM03, CM]. The latter react quickly to topology changes and keep routing overhead low in periods of little change. Therefore we detail existing work in the area of secure reactive routing protocols. We recall, first, generic functioning of a reactive routing protocol and then compare known secure reactive routing protocols for ad hoc networks.

Reactive (or on-demand) routing protocols [PR99, dym] enable dynamic, multi-hop routing between a set of participating nodes. They require messages to be exchanged only when a new (or non-existing) route between a source and a destination node

is to be discovered or when an existing route fails. In contrast, pro-active routing protocols require periodic exchange of routing information for loop-free and correct routes. Thus typically three kinds of messages can be defined for reactive routing protocols:

- **Route Request Message (RREQ)**: This is a message meant to discover a new route between a source and a destination node. The source node issues a RREQ message, specifying its own identity and that of the destination node (along with sequence numbers, etc.). Each intermediate node records the route to the source and forwards the message to its neighbors.
- **Route Reply Message (RREP)**: When the RREQ message reaches the destination node, it responds with a route reply message (RREP) towards the source node. Each intermediate node receiving this message records the route to the destination node and forwards the message to the next node on the route (established by RREQ message) to the source. When the source node receives the RREP message, routes have been established in both directions between the source and the destination node.
- **Route Error Message (RERR)**: This message is generated by a node when it receives a packet for a route that is no longer available. It is directed towards the source of the packet. Thus the source node can re-initiate route discovery by generating a RREQ message.

Examples of this kind of protocols include AODV [PR99], DSR [dsr] and DYMO [dym].

SRP : The Secure Routing Protocol (SRP) [PH02] is designed as an extension to existing reactive routing protocols and relies on the availability of a security association (SA) between the source node (S) and the destination node (T). An example of a SA is a secret symmetric key K_{ST} derived using the public keys of S and T. Thereafter S and T can authenticate each other's messages via this shared symmetric key using a MAC. The intermediate nodes that relay the RREQ towards the destination add

their IP address to the RREQ before forwarding but do not add any authentication information. Upon reception of a RREQ, the destination node verifies the integrity and authenticity of the RREQ by calculating the keyed hash of the request fields and comparing them with the MAC contained in the SRP header. If the RREQ is valid, the destination initiates a route replay (RREP) message wherein it puts the keyed hash of the accumulated path. The RREPs need to take the reverse route to get back to the source.

SRP suffers from the route cache poisoning attack: Any intermediate node that forwards the RREQ could add many false nodes to the message. This makes the probability of that route being selected as a route for T by S very low. This is because there is no point-to-point authentication of the messages. SRP is not immune to the wormhole attack.

ARIADNE : ARIADNE [HPJ02a] is an on-demand secure ad hoc routing protocol based on DSR that guarantees that the target node of a route discovery process can authenticate the initiator, that the initiator can authenticate each intermediate node on the path to the destination present in the RREP message and that no intermediate node can remove a previous node in the node list in the RREQ or RREP messages. The assumption is that each node shares a symmetric key with other nodes, and that each node has another TESLA [PCST00, PCST01] key chain whose one key is known to all others.

Thus ARIADNE prevents message tampering attacks but at a cost. The routing message size increases linearly with the route length as all the MACs are appended to the message. Also time synchronization is an essential requirement for the scheme to work.

ARAN : The ARAN [SDL+03] secure routing protocol is an on-demand routing protocol that provides point-to-point authentication of routing messages with the use of digital signatures. Thus each node which forwards the RREQ or RREP message verifies the previous node's signature, removes it if found valid and then adds its own signature. Thought the protocol is secure against message tampering, it is

prohibitively expensive to implement in ad hoc networks.

SAODV : The Secure AODV protocol [Zap] uses digital signatures to authenticate the non-mutable parts of any routing packet and *hash chains* to authenticate the mutable bits. In particular the field *hop count* of a packet is authenticated as follows: The source node generating the RREQ message, chooses a random number *seed* and puts $s = seed$ and $h^{MaxHopCount}(seed)$ into the RREQ packet. Any intermediate node re-broadcasting the RREQ message augments the *hop count* by one and replaces the previous hash value s by $h(s)$. To verify the authenticity of the current *hop count* k, any node can verify if $h^{MaxHopCount}(seed)$ equals $h^{MaxHopCount-k}(s)$. The protocol prevents nodes from decreasing the *hop count*, but not from keeping it same or increasing it, which can result in attacks as well.

endairA : The endairA protocol [BV04] uses digital signature verification on each hop of the route when the RREP message travels towards the source. The drawback is that the RREP message size depends linearly on the number of nodes in the route. The protocol assumes that wormhole attacks can be detected at the neighbour discovery level.

In Table 4.1 we compare secure reactive routing protocols. We mention the cryptographic mechanism employed by each protocol, resilience to tampering attacks (insertion or deletion of nodes in the path) and the computational and message size costs of implementing the security mechanism. Please note that the cost of a single asymmetric operation is usually much higher than that of a single symmetric key operation. The number of signature generations and verifications and the message size are for a route of n nodes. SRP uses only end-end authentication (only source and target nodes check the validity of the route request), and thus is vulnerable to tampering attacks by in between nodes. ARIADNE resists tampering attacks as each node verifies the earlier signature and appends its own, but requires loose time synchronization in the network. ARAN resists tampering attacks by requiring each node to verify the previous node's signature, and then putting its own signature after having removed the earlier one. This leads to large message sizes. SAODV requires the intermediate nodes to check the signature on the non-mutable fields of the message

and the use of Hash chains to resist tampering attacks. But it resists tampering attacks only partially with the use of hash chains. While nodes on a certain route cannot decrease the hop count of that route, they can still keep it same or increase it. endairA requires digital signature to be added to the RREP message at each hop, resulting in large message sizes. Later we show how a reactive routing protocol using our Aggregate Designated Verifier Signature scheme can achieve point-to-point authentication using (online) symmetric cryptographic operations and keep the routing message size constant. All are still prone to Wormhole attacks. The only known protocols to resist wormhole attacks are [PH03a] and [CY02]. While [PH03a] relies on transmission of messages across multiple routes to detect it, [CY02] relies on accurate global positioning systems.

In the next section we introduce a new cryptographic notion which can aid in point-to-point authentication of routing messages but relies on symmetric cryptography only. Also the space requirement of the authentication information does not depend on the number of nodes on the route. But it remains prone to wormhole attacks. This primitive can be used with any non-secure reactive routing protocol. It may be also be used in other applications where multiple messages need to be authenticated by the same single user.

4.2 Efficient Authentication for routing protocols

In this section we introduce a new cryptographic primitive to efficiently authenticate multiple messages sent by different entities to one single entity. We show how this cryptographic mechanism can be used to authenticate routing messages in routing protocols for ad hoc networks (especially reactive routing protocols). For proactive routing protocols, the scheme can only be used if it involves concurrent authentication of multiple messages from multiple senders by the same receiver. We first give below the preliminaries to understand the motivation and an efficient construction of this primitive.

	Cryptographic Mechanism	Resists Tampering	Computation		Message Size
			Gen	Ver	
SRP [PH02]	MAC	No	$O(1)$	$O(1)$	$O(1)$
ARIADNE [HPJ02a]	$SA + TESLA$	Yes	$O(n)$	$O(n)$	$O(n)$
ARAN [SDL+03]	DS	Yes	$O(n)$	$O(n)$	$O(1)$
SAODV [Zap]	DS + $Hash\ Chains$	No	$O(1)$	$O(n)$	$O(1)$
endairA [BV04]	DS	Yes	$O(n)$	$O(n)$	$O(n)$
Ours [BHL06, BHLar]	MAC	Yes	$O(n)$	$O(n)$	$O(1)$

SA: Shared symmetric key required between any two nodes.
DS: (Asymmetric) Digital Signatures.
Gen: Number of signature (symmetric or asymmetric) generations on the route
Ver: Number of signature (symmetric or asymmetric) verifications on the route

Table 4.1: Security Comparison of reactive routing protocols

4.2.1 Aggregate Signatures

Aggregate signatures were introduced in 2003 by Boneh, Gentry, Lynn and Shacham [BGLS03]. Basically, aggregating signatures means transforming n signatures on n distinct messages from n distinct users into a unique (shorter) signature. This unique signature is universally verifiable thanks to the signers' public keys. Thus a verifier can be provided with just one short signature rather than n, to verify whether the n users did indeed sign the n given messages. The verifier must still be provided n different public keys for verification. Also it is not possible to identify a misbehaving user in case of an invalid signature. Such signature schemes find applications in Public Key Infrastructure (PKI) setting and secure routing protocols like Secure Border Gateway Protocol: certificate chains of n signatures can be replaced by a single signature in the PKI setting; similarly, routing messages containing n signatures can be compressed to a single signature, significantly reducing the required transmission bandwidth.

An aggregate signature scheme consists of the following five algorithms:

KeyGen: For each user U, generates a pair of private and public keys, (sk_U, pk_U).

Sign: For a particular user U_i and given message $m \in \{0,1\}^*$, produces the signature σ on m under sk_{U_i}.

Verify: Given a user's public key pk_{U_i}, message m and a signature σ, returns 1 if σ is a valid signature on m for the given public key else returns 0.

Aggregate: Given a set of signatures σ_i on distinct messages m_i by user U_i respectively, provides a single aggregate signature σ.

Aggregate Verify: Given an aggregate signature σ along with a set of message-public key pairs (m_i, pk_{U_i}), returns 1 if all the given messages are distinct and σ is a valid aggregate signature on the messages m_i under the public keys pk_{U_i}, else returns 0.

Existing solutions include Boneh *et al.*'s pairing-based proposal [BGLS03] and Lysyanskaya *et al.*'s scheme [LMRS04], based on trapdoor permutations (like RSA). In the scheme of Boneh *et al.*, anyone holding the n signatures can aggregate them into one single aggregate signature. This scheme uses bilinear maps, which is quite an expensive operation (approximately equivalent to three modular exponentiations). While the scheme of Lysyanskaya*et al.* can only be aggregated by the signers as the aggregation operation and the signing operation are the same. An instantiation of their scheme with RSA requires ordering of the public moduli of the signers, which may not be easy to implement in practice.

Security of aggregate signatures requires an adversary to be not able to produce a valid aggregate signature on some messages of his choice, when he does not know the signing key (private key) of at least one of the signers. Thus the security is modeled as a game between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The adversary $\mathcal{A}$ is provided a single public key pk_U and is free to choose the other public keys. The adversary is also given access to a signing oracle on the challenge key pk_U. The goal of the adversary is to provide an existential forgery of an aggregate signature which includes the challenge key and a message (for this challenge key) for which it did not obtain the signature via the signing oracle. We do not give formal definition of aggregate signatures and discussion of the security model here (it can be found in [BGLS03]), as they are similar to the formal definitions of aggregate designated

verifier signatures, given in Section 4.2.4.

4.2.2 Designated Verifier Signatures

Designated verifier signatures are a non-interactive variant of the designated verifier proofs introduced in 1996 by Jakobsson, Sako and Impagliazzo [JSI96]. These non-interactive proofs have the property of being non-transferable, i.e., the designated verifier accepts a signature because he knows he had not produced it himself, but as he could have generated it on his own, he cannot convince anyone else. For the same reason, these signatures do not fit the most important property of traditional digital signatures, namely *non-repudiation* (therefore, calling them "signatures" is not strictly correct).

A designated verifier signature scheme is made up of the following four algorithms:

SKeyGen: It takes as input the public parameters and outputs a pair of public-private key pair (pk_A, sk_A) for the signer.

VKeyGen: It takes as input the public parameters and outputs a pair of public-private key pair (pk_B, sk_B) for the verifier.

DVS Sign: It takes as input a message m, a signing private key sk_A and a verifying public key pk_B and outputs a signature σ known as the B-designated verifier signature.

DVS Verify: It takes as input a bit string σ, a message m, a signing public key pk_A and a verifying private key sk_B and returns 1 if σ is a valid signature on m under the keys (pk_A, sk_B) else returns 0.

The security requirements are the following: the scheme must be existentially unforgeable under a chosen message attack and *source hiding*. The notion of source hiding means an attacker, even if he knows the private keys of both the signer and the verifier, must be (unconditionally) unable to determine who from the signer and the designated verifier produced a given signature. Another anonymity property capturing the concept of *strong* designated verifier signatures of Jakobsson *et al.* [JSI96]

is *privacy of signer's identity* [LV05]. It states that given two possible signers and a designated verifier signature, the adversary should not be able to tell which of the two signers actually produced the given signature, without the knowledge of the designated verifier's private key or the private key of one of the signers. For a formal definition and efficient proposals, we refer the reader to [SBWP03, SKM03, LV05].

4.2.3 Message Authentication Codes

A Message Authentication Code (MAC, for short) is a mechanism which provides mutual authentication between two users who share the same common private key. It consists of three algorithms:

$\mathsf{MAC.Key}$: it takes as input a security parameter k and returns a key space $\mathcal{K}$ and a particular key K.

$\mathsf{MAC.Gen}$: the MAC generator algorithm takes as inputs a message m and a key $K \in \mathcal{K}$, and outputs a string of bits $\tau \in \{0,1\}^{\ell}$ of fixed length.

$\mathsf{MAC.Verify}$: finally, the verification algorithm takes as inputs a message m, a key $K \in \mathcal{K}$ and a string of bits $\tau \in \{0,1\}^{\ell}$, and outputs 1 if τ is a valid MAC for m, or 0 otherwise.

Security of a MAC requires that an adversary is not able to forge a valid MAC without knowing the corresponding private key K, even if the adversary has access to MAC generation and verification oracles that he can query as he wants (adaptively). Formally, the following game $\mathtt{GAME}_1$ is played by a challenger and an attacker $\mathcal{A}$:

1. The challenger takes a security parameter $k \in \mathbb{N}$ and executes $K \leftarrow \mathsf{MAC.Key}(k)$. The key space $\mathcal{K}$ is given to $\mathcal{A}$.

2. The attacker $\mathcal{A}$ can make two kinds of queries:

   (i) a MAC generation query for message m_i; the challenger executes $\tau_i \leftarrow \mathsf{MAC.Gen}(m_i, K)$ and gives τ_i to $\mathcal{A}$.

(ii) a verification query for pairs (m_i, τ_i); the challenger returns to $\mathcal{A}$ the value $\mathsf{MAC.Verify}(m_i, K, \tau_i)$.

3. At the end, the attacker outputs a pair (m, τ).

Such an attacker $\mathcal{A}$ (t', ε')-breaks the unforgeability of the MAC if it runs in total time t' and it outputs with probability ε' a pair (m, τ) such that:

(i) message m has not been asked as a MAC generation query in step 2(i) of the game above; and

(ii) $\mathsf{MAC.Verify}(m, K, \tau) = 1$.

A MAC is (t', ε')-secure if there is no attacker $\mathcal{A}$ which (t', ε')-breaks the unforgeability of the MAC.

Besides this unforgeability consideration, message authentication codes also have anonymity properties. Following the terminology of [LV05], MACs achieve *source hiding*. This property is trivially obtained, as a common key is shared between two users : it is unconditionally infeasible for an attacker, even if he knows the secret key, to decide who from the two users produced a signature.

Security model for MACs is discussed in [BKR00].

4.2.4 Aggregate Designated Verifier Signatures

Having introduced the notions of Aggregate Signatures and Designated Verifier Signatures, we now introduce a new concept, namely: Aggregate Designated Verifier Signature. Essentially Aggregate Designated Verifier Signatures (or Ag_DVS for short) enable efficient aggregation of signatures meant for the same designated verifier. The motivation behind such schemes is as follows: Though aggregate signatures is a very useful concept, the only two existing schemes are either impractical or too expensive. It seemed that an additional structure as provided by trapdoor permutations or bilinear maps was an essential ingredient to be able to aggregate signatures. But

we show here that signatures can be very efficiently aggregated if they are all meant for the same designated verifier. And there exist many applications which have this feature (reactive routing protocols being one of them, as we show later). The Ag_DVS scheme that we propose requires only symmetric cryptography and thus is quite easy to run in ad hoc networks. Below we formally define an Ag_DVS scheme and present a new scheme along with its security analysis. Later in Section 4.3, we formalize the security model and analyze the security of the proposed scheme.

An aggregate designated verifier signature (Ag_DVS for short) scheme consists of the following seven algorithms:

Ag_DVS.Setup: it takes as input a security parameter k and outputs the *public parameters*. These public parameters are implicit inputs of all the following algorithms.

Ag_DVS.SKeyGen: it takes as input a security parameter k and returns a pair (sk_A, pk_A) of secret and public keys for the signer.

Ag_DVS.VKeyGen: it takes as input a security parameter k and returns a pair (sk_B, pk_B) of secret and public keys for the verifier.

Ag_DVS.Sign: it takes as inputs a message m, (possibly) the public key pk_B of the designated verifier B, and the secret key sk_A of the signer A. The output is a signature σ_{AB}.

Ag_DVS.Verify: the usual verification algorithm takes as inputs a message m, the public key pk_A of the signer, a signature σ_{AB} and the secret key sk_B of the designated verifier; it returns 1 if the verification is correct, and 0 if not.

Ag_DVS.Aggregate: this algorithm takes as input the public key pk_B of the verifier and n tuples[1] $\{(pk_{A_i}, m_i, \sigma_{A_iB})\}_{1\leq i\leq n}$ of messages correctly signed by different users A_i for the same designated verifier B. The output is an aggregate signature Σ_B.

Ag_DVS.Ag_Verify: finally, the algorithm to verify the correctness of an aggregate designated verifier signature takes as input the list of messages and public keys

[1]We suppose, for simplicity of the notation and without loss of generality, that each user has signed only one message m_i.

$\{(pk_{A_i}, m_i)\}_{1 \leq i \leq n}$, the aggregate signature Σ_B and the secret key sk_B of the verifier B. The output is 1 if the aggregate signature is correct and is 0 otherwise.

Constructing Ag_DVS from MACs

The mathematical framework for the construction is the following. We consider a group $G = \langle g \rangle$ of prime order q, generated by some element g; we consider multiplicative notation. We will suppose that the Computational Diffie-Hellman (CDH) problem is hard in this group.

Definition 4.1 *An algorithm $(t_{CDH}, \varepsilon_{CDH})$-solves the CDH problem if it receives as input a tuple (g, g^a, g^b) for random (and secret) values $a, b \in \mathbb{Z}_q^*$, and it outputs the value g^{ab} with probability ε_{CDH} and running time t_{CDH}.*

We assume that our group G is a $(t_{CDH}, \varepsilon_{CDH})$-hard group; that is, there is no algorithm which $(t_{CDH}, \varepsilon_{CDH})$-solves the CDH problem in G.

Our construction of the Ag_DVS scheme makes use of a MAC. Therefore, we will also assume that we have a (t', ε')-secure MAC, defined by the algorithms MAC.Key, MAC.Gen and MAC.Verify, with key space $\mathcal{K}$ and with fixed length ℓ for the MACs produced by MAC.Gen (in existing proposals of MACs, $\ell = 160$ is usually the case).

The Ag_DVS scheme that we propose works as follows:

Ag_DVS.Setup: if the security parameter is k, then a $(t_{CDH}, \varepsilon_{CDH})$-hard group $G = \langle g \rangle$ is chosen such that the order q of G is a 2^k-bit prime number. A cryptographic hash function $H : G \to \mathcal{K}$ is also chosen and published.

Ag_DVS.SKeyGen/DVS.VKeyGen: for each user U, a random number $sk_U \in \mathbb{Z}_q^*$ is chosen, and the matching public key is defined to be $pk_U = g^{sk_U}$.

Ag_DVS.Sign: given a message m, the public key pk_B of a designated verifier B, and the secret key sk_A of a signer A, the signature σ_{AB} is defined to be

$$\sigma_{AB} = \mathsf{MAC.Gen}\big(m, H(pk_B^{sk_A})\big) \ \in \{0,1\}^\ell.$$

Ag_DVS.Verify: the standard verification algorithm takes as inputs a message m, the public key pk_A of the signer, a signature σ_{AB} and the secret key sk_B of the designated verifier. It returns the bit output by

$$\mathsf{MAC.Verify}\big(m, H(pk_A^{sk_B}), \sigma_{AB}\big).$$

Note that the result of the verification is the correct one, since $pk_A^{sk_B} = pk_B^{sk_A} = g^{sk_A sk_B}$.

Ag_DVS.Aggregate: the input consists of n tuples $\{\ (pk_{A_i}, m_i, \sigma_{A_iB})\ \}_{1\leq i\leq n}$. The output is the value

$$\Sigma_B = \bigoplus_{1\leq i\leq n} \sigma_{A_iB}\ \in \{0,1\}^{\ell}.$$

Ag_DVS.Ag_Verify: this algorithm takes as input the list of messages and public keys $\{(pk_{A_i}, m_i)\}_{1\leq i\leq n}$, the aggregate signature Σ_B and the secret key sk_B of the verifier B. The output is 1 if and only if

$$\Sigma_B = \bigoplus_{1\leq i\leq n} \mathsf{MAC.Gen}\big(m_i, H(pk_{A_i}^{sk_B})\big).$$

Note that, roughly speaking, the verifier B must re-calculate all the aggregated signatures σ_{A_iB} to verify the correctness of an aggregate (constant-length) signature Σ_B.

The most costly operations in both signing and verifying consist of modular exponentiations, namely computing $pk_B^{sk_{A_i}}$ for signing and $pk_{A_i}^{sk_B}$ for verifying. These operations do not depend on the specific signed message(s), and so can performed off-line, reducing therefore the on-line required operations to a MAC generation or verification.

4.3 Security Analysis

In this section, we present the security model to analyze the security of our scheme and then present a proof of security of the latter.

4.3.1 Security Model for Ag_DVS Schemes

We consider the highest level of security for an Ag_DVS scheme, which is obtained by combining the security models for standard designated verifier signature schemes and aggregate signature schemes. The idea is that an attacker $\mathcal{F}$ cannot obtain a valid aggregate signature for some designated verifier B, if he does not know the secret key sk_B of B, or all the secret keys sk_{A_i} of the authors of the aggregated signatures, or the aggregated signatures themselves. The attacker is allowed to make signature and verification queries. Formally, the security is defined by the following game $\texttt{GAME}_2$ that the attacker $\mathcal{F}$ plays against a challenger:

1. The challenger takes a security parameter k and executes $\mathsf{Ag\_DVS.Setup}(k)$, $(sk_{A_1}, pk_{A_1}) \leftarrow \mathsf{Ag\_DVS.SKeyGen}(k)$ and $(sk_B, pk_B) \leftarrow \mathsf{Ag\_DVS.VKeyGen}(k)$.

2. The attacker $\mathcal{F}$ receives the public keys pk_{A_1} and pk_B. He can execute the protocols $\mathsf{Ag\_DVS.SKeyGen}$ and $\mathsf{Ag\_DVS.VKeyGen}$ by himself to obtain other pairs of secret and public keys.

3. The attacker $\mathcal{F}$ can make three kinds of queries:

   (i) a signature query for message $\tilde{m}$; the challenger executes $\tilde{\sigma}_{A_1B} \leftarrow \mathsf{Ag\_DVS.Sign}(\tilde{m}, pk_B, sk_{A_1})$ and gives $\tilde{\sigma}_{A_1B}$ to $\mathcal{F}$.

   (ii) a verification query for pairs $(\tilde{m}, \tilde{\sigma}_{A_1B})$ of his choice; the challenger returns to $\mathcal{F}$ the value $\mathsf{Ag\_DVS.Verify}(\tilde{m}, pk_{A_1}, \tilde{\sigma}_{A_1B}, sk_B)$.

   (iii) a verification query for an aggregate signature $\tilde{\Sigma}_B$ corresponding to a list of pairs $\{(\tilde{pk}_{A_i}, \tilde{m}_i)\}$ of his choice; the challenger returns to $\mathcal{F}$ the value $\mathsf{Ag\_DVS.Ag\_Verify}(\{(\tilde{pk}_{A_i}, \tilde{m}_i)\}, \tilde{\Sigma}_B, sk_B)$.

4. At the end, the attacker $\mathcal{F}$ outputs an aggregate signature Σ_B for a list of pairs $L = \{(pk_{A_i}, m_i)\}_{1 \leq i \leq n}$.

The adversary $\mathcal{F}$ succeeds if:

(i) $\mathsf{Ag\_DVS.Ag\_Verify}\big(pk_{A_1}, m_1, \ldots, pk_{A_n}, m_n, \Sigma_B, sk_B\big) = 1$; and

(ii) there is at least one pair $(pk_{A_1}, m_1) \in L$ such that m_1 has not been asked by $\mathcal{F}$ as a signature query in step 3(i) of the game above.

If the running time of $\mathcal{F}$ is t and its success probability is ε, then we say that it (t, ε)-breaks the unforgeability of the Ag_DVS scheme.

Definition 4.2 *An Ag_DVS scheme is* (t, ε)-secure *if there is no attacker* $\mathcal{F}$ *which* (t, ε)*-breaks the unforgeability of this scheme.*

The property of source hiding for standard designated verifier signature schemes can be extended to our scenario of Ag_DVS schemes.

Definition 4.3 (source hiding) *An Ag_DVS scheme is* source hiding *if there exists an algorithm* Simul *which takes as input only the secret key of the designated verifier and which produces bit strings which are indistinguishable (even with the knowledge of all secret keys) from the distribution of real aggregate signatures produced by real signers.*

To prove the property of source hiding, it is sufficient to exhibit such an algorithm.

4.3.2 Security Proof

We are going to prove that the Ag_DVS scheme constructed in the previous section is secure, provided the employed MAC is secure and the CDH problem is hard in

the group G. The proof is in the random oracle model [BR93] for the hash function $H : G \rightarrow \mathcal{K}$. This means we assume in the proof that this function behaves as a totally random function; any attacker has access to an oracle which gives consistent answers, simulating the ideal behavior of the function H.

Theorem 4.1 *If the employed MAC is (t', ε')-secure and G is a $(t_{CDH}, \varepsilon_{CDH})$-hard group, then the proposed Ag_DVS scheme is (t, ε)-secure, where*

$$\varepsilon = \varepsilon' + \varepsilon_{CDH} \quad \textit{and} \quad t = \min\{t' - Q_H, t_{CDH} - Q_H\},$$

where Q_H is the total number of queries that an attacker $\mathcal{F}$ against the Ag_DVS scheme can make to the random oracle which models H.

Proof: Let us assume, to the contrary, that there exists some attacker $\mathcal{F}$ which (t, ε)-breaks the unforgeability of our Ag_DVS scheme. We are going to use $\mathcal{F}$ as a subroutine to construct an attacker $\mathcal{A}$ which breaks the unforgeability of the employed MAC. Let us explain how this attacker $\mathcal{A}$ plays the corresponding game $\texttt{GAME}_1$ (recall Section 4.2.3).

The challenger in game $\texttt{GAME}_1$ takes a security parameter k and executes $K \leftarrow \mathsf{MAC.Key}(k)$. The key space $\mathcal{K}$ is given to $\mathcal{A}$ (attacker in game $\texttt{GAME}_1$).

At this point, $\mathcal{A}$ initiates the running of the attacker $\mathcal{F}$ which is assumed to exist against the Ag_DVS scheme. $\mathcal{A}$ is going to play the role of a challenger against $\mathcal{F}$ in the game $\texttt{GAME}_2$ (recall Section 4.3).

1. First of all, $\mathcal{A}$ takes a security parameter k' and executes $\mathsf{Ag\_DVS.Setup}(k')$, $(sk_{A_1}, pk_{A_1}) \leftarrow \mathsf{Ag\_DVS.SKeyGen}(k')$ and $(sk_B, pk_B) \leftarrow \mathsf{Ag\_DVS.VKeyGen}(k')$.

2. $\mathcal{A}$ sends to $\mathcal{F}$ the public keys pk_{A_1} and pk_B.

3. The attacker $\mathcal{F}$ can make in this case four kinds of queries (because we are in the random oracle model):

(i) a signature query for message $\tilde{m}$; to answer this query, attacker $\mathcal{A}$ asks its own oracle (recall definition of $\texttt{GAME}_1$) the MAC of the message $\tilde{m}$ with respect to the unknown key K. The obtained MAC value $\tilde{\sigma}_{A_1B} = \texttt{MAC.Gen}(\tilde{m}, K)$ is transferred to $\mathcal{F}$; note that the attacker $\mathcal{A}$ is implicitly imposing the relation $K = H\big(pk_B^{sk_{A_1}}\big) = H\big(g^{sk_B sk_{A_1}}\big)$.

(ii) a verification query for pairs $(\tilde{m}, \tilde{\sigma}_{A_1B})$; in this case, attacker $\mathcal{A}$ can again query its own MAC verification oracle, and return to $\mathcal{F}$ the bit that it obtains from $\mathsf{MAC.Verify}(\tilde{m}, K, \tilde{\sigma}_{A_1B})$.

(iii) a verification query for an aggregate signature $\tilde{\Sigma}_B$ corresponding to a list of pairs $\{(\tilde{pk}_{A_i}, \tilde{m}_i)\}_{1\leq i\leq n}$ of his choice; in this case, attacker $\mathcal{A}$ can use the knowledge of the secret key sk_B to compute

$$\tilde{\Sigma}'_B = \bigoplus_{2\leq i\leq n} \mathsf{MAC.Gen}\big(\tilde{m}_i, H(\tilde{pk}_{A_i}^{sk_B})\big)$$

and then compute

$$\tilde{\sigma}_{A_1B} = \tilde{\Sigma}'_B \bigoplus \tilde{\Sigma}_B$$

and returns to $\mathcal{F}$ the bit that it obtains from $\mathsf{MAC.Verify}(\tilde{m}, K, \tilde{\sigma}_{A_1B})$.

(iv) a hash query $H(\tilde{y})$ for some element $\tilde{y} \in G$; to answer these queries, attacker $\mathcal{A}$ maintains a table of values $\texttt{TAB}$. Then it proceeds as follows: if $\tilde{y} = g^{sk_B sk_{A_1}}$, then the attacker $\mathcal{A}$ stops. Otherwise, if looks for $\tilde{y}$ in $\texttt{TAB}$; if the value is already there, then $\mathcal{A}$ sends to $\mathcal{F}$ the value $H(\tilde{y})$ stored in $\texttt{TAB}$. If the value $\tilde{y}$ is new, then $\mathcal{A}$ chooses at random the value $H(\tilde{y})$ in $\mathcal{K}$, sends it to $\mathcal{F}$ and stores the new relation in $\texttt{TAB}$.

Suppose that $\mathcal{F}$ asks the query $H(g^{sk_B sk_{A_1}})$, with a probability which is greater than ε_{CDH}. Then we have that the algorithm $\mathcal{F}$ can solve the CDH problem in time $t + Q_H \leq t_{CDH}$ and with probability greater than ε_{CDH}. This gives us a contradiction with the fact that the group G is assumed to be $(t_{CDH}, \varepsilon_{CDH})$-hard.

We can thus assume that $\mathcal{F}$ asks the query $H(g^{sk_B sk_{A_1}})$ with probability less

than ε_{CDH}. This means that the attacker $\mathcal{A}$ does not stop in the queries' phase with probability at least $1-\varepsilon_{CDH}$. In this case, the environment of the attacker $\mathcal{F}$ has been perfectly simulated by $\mathcal{A}$.

4. Therefore, by hypothesis, the attacker $\mathcal{F}$ outputs in total time t and with probability ε a correct aggregate signature Σ_B for a list of pairs $L = \{(pk_{A_i}, m_i)\}_{1\leq i\leq n}$ such that there exists a pair (pk_{A_1}, m_1) satisfying that m_1 has not been queried by $\mathcal{F}$ to the Ag_DVS.Sign oracle, in step 3(i).

   For the rest of pairs (pk_{A_i}, m_i), for $i = 2, \ldots, n$, the attacker $\mathcal{A}$ computes the values $\sigma_{A_iB}^{(i)} = \mathsf{MAC.Gen}\big(m_i, H(pk_{A_i}^{sk_B})\big)$ by using his knowledge of the secret key sk_B of B, and then he computes the value

$$\sigma_{A_1B}^{(1)} = \Sigma_B \oplus \bigoplus_{2\leq i\leq n} \sigma_{A_iB}^{(i)}.$$

   By definition, this is a valid designated verifier signature for the message m_1 and signer A_1, satisfying the MAC verification equation with private key $H\left(g^{sk_B sk_{A_1}}\right) = K$ (implicit equality).

Summing up, the attacker $\mathcal{A}$ obtains in time at most $t + Q_H \leq t'$ a valid forgery of the employed MAC (note that $\mathcal{A}$ has never queried the message m_1 to its MAC.Gen oracle). The total success probability of $\mathcal{A}$ is the probability that: $\mathcal{A}$ does not stop and $\mathcal{F}$ outputs a valid forgery for the Ag_DVS scheme. This probability is thus greater than $(1-\varepsilon_{CDH})\varepsilon \geq \varepsilon - \varepsilon_{CDH} = \varepsilon'$.

Therefore, we have constructed an attacker $\mathcal{A}$ which breaks the unforgeability of the MAC in time less than t' and with success probability greater than ε'. This contradicts the fact that the employed MAC is assumed to be (t', ε')-secure.

This contradiction leads to the desired result: there cannot exist an attacker which (t, ε)-breaks the unforgeability of our Ag_DVS scheme.

As a result of this theorem, we have that the probability of breaking our scheme will be always $\varepsilon \leq \varepsilon' + \varepsilon_{CDH}$. Therefore, the scheme will be secure (small value of ε) if

the employed MAC is secure (small value of ε') and the CDH problem is really hard to solve in the group G (small value of ε_{CDH}).

Theorem 4.2 *The proposed Ag_DVS scheme is unconditionally source hiding.*

Proof: This is trivial as the designated verifier shares a key $g^{sk_B sk_{A_i}}$ with each signer A_i and therefore can produce all the MACs by himself.

In [LWB05], Lipmaa, Wang and Bao gave a stronger model for DVS to prevent the signer and the designated verifier to delegate their signing/verifying capability without revealing their secret key. Our scheme is not secure in this model since, if A (or B) reveals $pk_B^{sk_A}$ (or $pk_A^{sk_B}$), then any user can impersonate them. Anyway, we want to stress that this situation is not relevant in our context, because if A (or B) reveals this value, he will be one of the victims of the possible attack.

4.4 Application to Authenticated Routing

We explain with an example below how the above scheme can be used to secure reactive routing protocols. Consider Figure 4.1, where node B generates a RREQ message when it wants to find a route to node A_5. The message is forwarded (as it is or slightly modified depending on the protocol) as shown in the figure. When the message reaches the destination node A_5, it replies with a RREP message. Our Aggregate Designated Verifier Signature scheme of previous section enables us to include authentication information from all the nodes from the source to the destination. Each node who claims to be on the route to node A_5 needs to provide a suitable series of messages and an aggregate designated verifier signature on them to convince node B. Thus node B is convinced that there exists a route from node B to node A_5 via nodes A_1 and A_4 if he can obtain messages from each node saying their position in the route and an aggregate signature to confirm the authenticity of all these messages. Thus as shown in Figure 4.2, when node A_5 generates the RREP

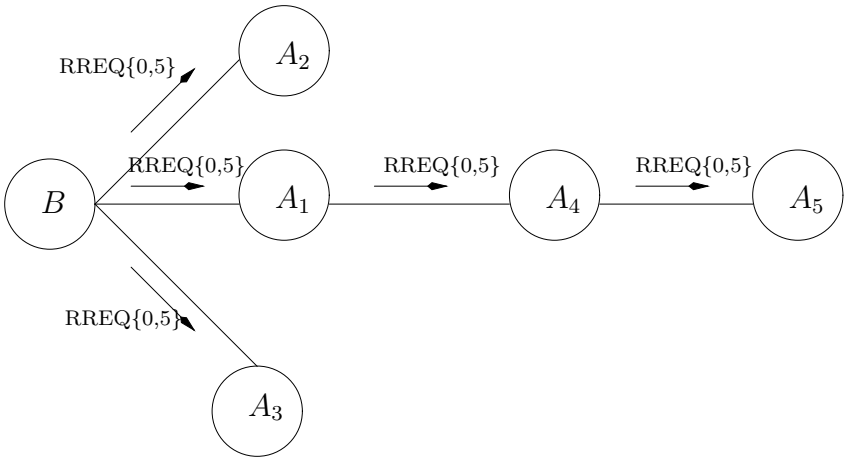
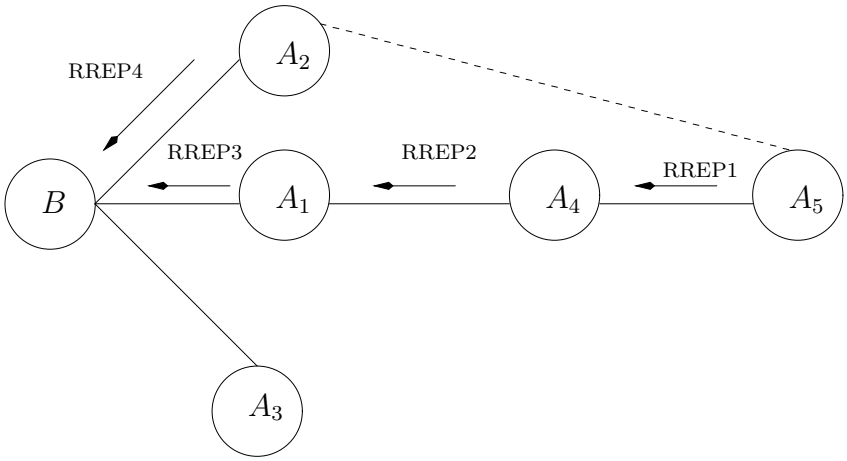


Figure 4.1: Route Request Example

message, it also appends its signature σ_{A_5B} on that message, which can only be verified by node B. On receiving this RREP message, node A_4 adds his own message, m_4 and XORs its signature σ_{A_4B} to σ_{A_5B} and appends it. The same is done by node A_1. On receiving the message RREP3, node B checks for messages m_5, m_4, m_1 and an aggregate signature $\sigma_{A_1B} \bigoplus \sigma_{A_4B} \bigoplus \sigma_{A_5B}$ on them. What the messages actually contain depends on the particular protocol used. It could be the number of hops (as in AODV) or the route path (as in source routing). And also it may be possible to construct messages m_i from just one single message logically.

If a malicious node (say A_2) wants to advertise a false route to node A_5, it would have to produce an aggregate signature $\sigma_{A_2B} \bigoplus \sigma_{A_5B}$, which is impossible for node A_2 to produce.

Authentication of RREQ messages could also be done for source routing protocols. A drawback of the solution we propose is that designated verifier signatures do not achieve non-repudiation. Intuitively, given a designated verifier signature, both the real signer and the verifier could have produced this signature. In the context of reactive routing it implies that a node B who has found a valid and authenticated route until some node A_j, cannot convince anyone else of this fact, because it could have produced the aggregate designated verifier signature by itself. Therefore, if a different node C wants to obtain a valid route until A_j and it reaches node B in this



RREP1 = { m_5 , σ_{A_5B} }
RREP2 = { m_5, m_4, $\sigma_{A_4B} \bigoplus \sigma_{A_5B}$ }
RREP3 = { m_5, m_4, m_1, $\sigma_{A_1B} \bigoplus \sigma_{A_4B} \bigoplus \sigma_{A_5B}$ }
RREP4 = { m_5, m_2, $\sigma_{A_2B} \bigoplus \sigma_{A_5B}$ }

Figure 4.2: Route Reply Example

process, node B cannot convince C that there exists a valid route $B - A_j$ by using his aggregate signature. As a consequence, the nodes in the path $B - A_j$ must sign a message again, this time with node C as designated verifier. This results in an efficiency loss if the nodes are all honest, but on the contrary it is a positive point in order to prevent a malicious node B' to cheat by claiming that there exists a valid path $B' - A_j$ when it actually does not exist.

Therefore, it is not clear whether the public verification property achieved by standard aggregate signatures (with public verification) will lead to a real efficiency improvement in practice. In fact, in mobile ad-hoc networks where nodes change their position very often, the availability of routes must be almost constantly updated, and so it is difficult to imagine an "old" route $B - A_j$ to be accepted by a node C in order to construct a new route $C - A_j$.

4.5 Conclusion

The clear advantage of our scheme is that the RREQ/RREP message size does not depend on the number of nodes on the route. As at each node, the new signature is XORed with the existing aggregated signature. The use of ADVS can not only prevent message tampering attacks but also prevent replay attacks and rushing attacks (see Section 2.3.3) by making appropriate use of time-stamps, nonces or sequence numbers (depending on the exact reactive routing protocol being used). As mentioned earlier, the use of ADVS does not prevent wormhole attacks. Currently only non-cryptographic mechanisms (like use of Global Positioning System) are known to help in prevention of such attacks.

Chapter 5

Conclusions and Future Work

Securing ad hoc networks is a challenging task. Lack of commonly trusted authorities, limited computational abilities of the nodes, limited bandwidth and dynamic nature of the network makes the task of securing ad hoc networks more challenging than in a traditional network. At the same time, there is an increased need for nodes to cooperate between themselves to be able to access services on the network. Energy constrained devices with sporadic connectivity to the network means all security services must be distributed so that the failure of a single node does not handicap the network.

Research in security of ad hoc networks has focussed on all basic security services including Access Control, Authentication, Key management and Confidentiality. Besides that a very intense area of research has been security of routing protocols in such networks. While in the Internet, the function of routing is performed by selected nodes which are often the most protected (physically as well as logistically) nodes of the network, in ad hoc networks this functionality needs to be performed by all the nodes. Thus the very basic functioning of the network could possibly depend on a mobile palm-top with limited energy and no physical protection. Thus the increased attention to secure routing in ad hoc networks.

In this thesis, we address two of the security areas mentioned above, namely, key

management and secure routing. Both are very basic security services required to enable other security services.

We explore in this thesis why group key agreement protocols are the most suitable of key management techniques in ad hoc networks. Group key agreement protocols provide the ability to derive shared secret keys from the contributions of all the nodes in a dynamic ad hoc networks. Some existing GKA protocols require intensive computational operations to be performed by each node, some others require multiple message exchanges. This makes these protocols unsuitable for the constrained devices (with limited bandwidth) often present in an ad hoc network. Existing efficient protocols, with procedures to handle network dynamism, exist but require a well-structured network to be able to perform. By a well-structured network, we mean a network, which has a logical ordering of the participants. This is required to able to effectuate synchronized communication required in such protocols. Again this is a problem to achieve in ad hoc networks because of the lossy nature of the medium as well as the time-varying composition of the network. Thus such protocols, requiring synchronized communication, can be possibly brought to a halt due to the failure of a single node. We propose a new group key agreement protocol, efficient in the number of communication rounds and computational effort required (by each node). They do not require a well-structured network to perform and thus failure of few nodes does not prevent other nodes from calculating the shared secret key. It does require one node in the network to perform different functions than the other nodes in the network. Such a node could be chosen by application-specific criterion or an auto-election mechanism that we propose. Again this "special" node can be changed easily and as often as required, thus distributing this task all around the network. The protocol outperforms most existing protocols in its simplicity and efficiency. We prove it secure in the standard model and present results of our efforts to implement this protocol in a real scenario.

Possible areas of future work include the study of the effect of compromised nodes on the secrecy of the shared secret keys, and the enhancement of the security model to "concretely" capture the presence of compromised nodes and lossy nature of the

communication medium. The few proposed protocols which deal with compromised nodes (or malicious insiders), try to prevent "key control" by one of these participants. And the usual strategy adopted is to add a round of key confirmation whereby each participant of the GKA protocol is assured of the value of the secret keys derived by other participants. It remains to study the effect of compromised nodes across concurrent executions and incomplete executions of the GKA protocol. The security model, thus needs to be strengthened to take into account such malicious insiders.

The second part of the thesis deals with the issue of secure routing in ad hoc networks. Existing attempts to secure routing in ad hoc networks have either employed "light-weight security" mechanisms (like end-end authentication of routing messages or *hash chains* for node-node authentication) to achieve efficiency or "heavy-duty security" mechanisms (like node-node signature verification and authentication) to achieve higher levels of security. While the routing protocols using the former security approach are vulnerable to a whole range of attacks, protocols using the latter approach add a heavy computational burden on each node and significantly increase the size of the routing messages. We introduce a new cryptographic notion: *Aggregate Designated Verifier Signatures*, which allow multiple signatures destined for the same single entity to be efficiently aggregated. Such a scheme allows to achieve strong node-node authentication of the routing messages but at the same time maintains a constant (routing) message size as well as does not require intensive computational effort by each node. The constraint is that, such a scheme works when the destination of the multiple routing messages is the same. That implies, that, while this scheme can be very efficiently adopted for reactive routing protocols, it may not always work in the case of non-reactive routing protocols. In the latter case *Aggregate Signatures* might be a better choice. But the only existing practical aggregate signature scheme is computationally demanding. We provide a security model to study the security of ADVS schemes. We propose a ADVS scheme, that is simple, less computation-intensive and provably secure in this model. Possible area of future work is again to strengthen the security model so as to effectively capture the issues (malicious participants, lossy nature of the medium) of ad hoc networks.

Appendix A

Résumé en français

Introduction

Les réseaux ad hoc sont des réseaux sans infrastructure et sont formés par un ensemble de dispositifs mobiles communiquant dans un milieu sans fil. En raison de l'absence d'infrastructure fixe, toutes les tâches d'installation du réseau et de sa maintenance doivent être exécutés par les nœuds eux-mêmes. Les réseaux ad hoc où tous les nœuds sont à proximité immédiate (dans la gamme de transmission) les uns des autres sont désignés sous le nom des réseaux à un saut (one-hop), alors que des réseaux où un nœud doit se reposer sur des nœuds intermédiaires pour atteindre d'autres nœuds sont désignés sous le nom des réseaux multi-sauts (multi-hop). Dans ce dernier cas, le routage des messages est une fonctionnalité principale qui doit être exécutée par (potentiellement) tous les nœuds dans le réseau. Puisque les nœuds ont une ressource d'énergie limitée, toutes les fonctionnalités doivent être efficaces en utilisation d'énergie, ce qui implique à la fois de l'efficacité en termes de communications mais aussi en termes informatiques. En outre les épuisements de batterie peuvent faire disparaître des nœuds du réseau de manière abrupte, causant des changements soudains dans la composition du réseau. La mobilité des nœuds est un autre facteur contribuant à la composition changeante rapide du réseau. Le milieu sans fil lui-même

est moins fiable ce qui complique ainsi le problème. Le rapport entre les nœuds dans un réseau ad hoc est lui-même tout à fait différent de celui des nœuds dans un réseau câblé traditionnel. Dans ce dernier, chaque nœud simple (au moins chaque nœud de bord) se fonde essentiellement sur un nœud donné (le passage) pour communiquer et pour accéder à des services sur d'autres nœuds. Ce nœud indiqué est considéré de confiance et change rarement. Tandis que dans les réseaux ad hoc, les nœuds doivent coopérer avec leurs voisins (qui comptent à leur tour sur leurs voisins) pour àccéder à des services divers. Les voisins de n'importe quel nœud donné peuvent changer souvent et cela exige l'établissement de nouveaux rapports de confiance. Ainsi il y a un besoin inhérent de coopération et de communication du groupe dans un réseau ad hoc.

Les réseaux ad hoc ont été longtemps considérés de valeur critique aux opérations et aux scénarios militaires sur le champ de bataille. L'exemple le plus ancien d'un réseau ad hoc mobile est celui de réseau radio de paquets. Ils sont maintenant vus comme alternative attrayante aux réseaux cablés dans beaucoup d'autres scénarios comprenant la gestion de catastrophe, les réseaux véhiculaires, les réseaux personnels de secteur et les applications de collaboration de réseau.

À cause de la nature d'émission du milieu sans fil, la communication dans les réseaux ad hoc prête le flanc à l'écoute clandestine. En outre, le manque d'un mécanisme physique de contrôle d'accès permet à n'importe quel nœud de faire partie du réseau (souvent sans être détecté). La basse sécurité physique des nœuds signifie aussi que certains des nœuds dans le réseau peuvent potentiellement être compromis. Ainsi l'utilisation des mécanismes cryptographiques est nécessaire pour réaliser des communications sûres et pour fournir des services fiables dans de tels réseaux. Tous les mécanismes cryptographiques de base pour réaliser la confidentialité, l'intégrité, l'authentification, la non-repudiation et la disponibilité se fondent sur des clefs *cryptographiques*. De ce fait la fabrication de clefs disponibles et leur maintenance (gestion principale) dans de tels réseaux dynamiques avec une infrastructure minimale est une condition de base. La tâche de la gestion des clés est compliquée par les ressources limitées (largeur de bande, puissance, calcul) au niveau des nœuds, et

aussi par le manque d'autorité de confiance centrale. Fournie les mécanismes efficaces d'authentification en l'absence d'une autorité de confiance est un grand défi en lui-même. Le besoin de routage sécurisé dans les réseaux ad hoc est beaucoup plus accentué que dans les réseaux câblés en raison de la participation de presque tous les nœuds au processus de routage. En effet, dans les réseaux cablés, la fonction du cheminement est exécutée par quelques nœuds choisis, qui sont souvent fixés par des mécanismes physiques et cryptographiques. Ainsi en bref, les caractéristiques des réseaux ad hoc rendent beaucoup de mécanismes cryptographiques existants (y compris des protocoles de gestion d'authentification et de clef) inopérants et exigent de modifier beaucoup d'autres mécanismes avant qu'ils puissent être utilisés dans de tels réseaux.

Dans cette thèse nous présentons les problèmes liés aux réseaux ad hoc mobiles, et nous contribuons en proposant une nouvelle primitive et des protocoles cryptographiques bien adaptés aux réseaux ad hoc. En particulier, nous nous concentrons sur les deux secteurs fondamentaux de l'établissement de cléf et de la sécurisation de routage.

Le premier contribution de cette dissertation est un nouveau protocole d'accord de clef de groupe qui permet aux participants de convenir d'une clef secrète, qui peut être employée plus tard pour d'autres mécanismes cryptographiques. Le protocole est simple, contributif, robuste (aux échecs de nœuds et de liens) et ne fait l'hyppothèse d'aucune information sur la topologie du réseau. Nous présentons le version sans authentification et avec authentification du protocole, avec des preuves de sécurité dans le modèle standard, utilisé pour de tels protocoles. Nous fournissons également des détails d'exécution pour utiliser un tel protocole dans un réseau ad hoc typique.

Le deuxième contribution de cette dissertation est l'introduction de la notion d'aggrégation de signatures à vérificateur designée. Celles-ci sont essentiellement des signatures à vérificateur designées qui permettent une agrégation efficace. Nous fournissons des définitions formelles et un modèle de sécurité pour de telles signatures. Nous montrons alors comment de telles signatures pourraient être une primitive très utile pour authentifier des messages dans des protocoles de routage (particulièrement dans les

protocoles réactifs de routage). Ces signatures permettent l'authentification de messages de routage dans les réseaux ad hoc d'une facon plus efficace que les schémas existants.

Cette thèse est organisée comme suit : la section "Sécurité dans réseaux ad hoc" est un résumé des chapitres 1 et 2 de la thèse et constitue une introduction aux réseaux ad hoc et à leur problèmes de sécurité. La section "Létablissement de clefs dans les réseaux ad hoc" introduit les travaux existants pour établir la clé dans un réseau, et étudie leur faisabilité dans un réseau ad hoc. Nous proposons un nouveau protocole d'accord de clef de groupe qui est particulièrement approprié aux réseaux ad hoc, mais qui surpasse également la plupart des protocoles connus dans les réseaux traditionnels. Dans la section "sécurisation du routage dans un réseau ad hoc" nous présentons une nouvelle primitive cryptographique, que nous appelons une signature à vérificateur designée agregée (ADVS), qui permet l'agrégation efficace des signatures multiples à l'intention d'un mm̂e vérificateur. Nous présentons un exemple concret de cette primitive et nous montrons comment cette primitive peut être très efficacement employée pour sécurisér des protocoles réactifs de routage dans les réseaux ad hoc. Nous concluons dans la section "Conclusions et Perspectives" avec quelques perspectives dans ces domaines, pour des travaux futurs.

Sécurité dans réseaux ad hoc (Chapitre 1 et 2)

Réseaux ad hoc

Les réseaux ad hoc ont beaucoup de traits qui les rendent tout à fait distincts des réseaux câblé, et donc exigent ainsi des manières innovatrices de mettre en application les fonctionnalités de réseau. Nous discutons ci-dessous certains de ces caractères qui distinguent les réseaux ad hoc.

1. Support de transmission sans fil : le moyen sans fil employé par les nœuds pour communiquer avec l'un l'autre est variable avec le temps, et les liens peuvent être asymétriques. Il est aussi moins fiable et plus enclin à l'interférence comparée

à un milieu câblé.

2. Mobilité : on ne s'attend pas à ce que les nœuds dans de tels réseaux soient fixés à un certain endroit, *i.e.*, une adresse de destination fixe n'implique pas un endroit fixe de destination. Par conséquent, si un nœud était accessible dans la dernière transmission, cela ne implique pas qu'il sera accessible (le cas échéant) exactement de la même manière la prochaine fois. Ainsi, les réseaux simples à un saut peuvent exiger des nœuds d'ajuster dynamiquement la gamme de leurs transmissions tandis que les réseaux à multi-sauts doivent avoir des protocoles dynamiques de routage.

3. Gestion de puissance : comme les nœuds ne sont pas fixés, ils utilisent des batteries en tant que source d'énergie. Ainsi les mécanismes et les protocoles conçus pour de tels réseaux doivent respecter les contraintes d'énergie.

4. Pair-a-pair : il n'y a pas de nœud fixé avec des rôles prédéfinis. Ainsi, tous les protocoles doivent être conçus pour les environnements distribués, composés des pairs et doivent être assez robustes pour manipuler des topologies dynamiques distribuées.

Ces différentes caractéristiques des réseaux ad hoc sans fil exigent des techniques diffentes que pour les réseaux câblés, particulièrement au niveau des trois couches les plus basses, pour exécuter efficacement les fonctions de réseau. Les normes largement adoptées pour les réseaux sans fil, à la couche physique et à la couche des liens, incluent IEEE 802.11 (pour les réseaux locaux sans fil) et IEEE 802.15 (pour les réseaux personnels sans fil). La norme 802.11 fournit des caractéristiques de couche secondaire MAC et de couche physique pour LANs sans fil. Le groupe de travail de 802.15 WPAN [802] se concentre sur des normes pour les réseaux personnels ou les réseaux courts de radio de distance. La gestion de réseau sans fil des WPANs addresse des dispositifs de calcul portatifs et mobiles tels que des ordinateurs portables, aides personnels numériques (PDAs), périphériques, téléphones cellulaires, pageurs et électronique grand public, permettant à ces dispositifs de communiquer et interagir entre eux. Le groupe de travail d'IETF MANET (réseaux Ad hoc mobiles) regarde

les protocoles efficaces de routage au niveau de la couche réseau, qui sont légers et bien adaptés à la dynamicité du réseau. Dans ce groupe existent aussi quelques propositions pour des protocoles de couche transport pour les réseaux ad hoc.

IEEE 802.11 - IEEE 802.11 [i80] est une de normes de l'IEEE 802 (vue d'ensemble et architecture de réseaux de zone métropolitaine et réseaux locaux) traitant les LANs sans fil (WLANs). La topologie de base d'un WLAN s'appelle un BSS (service de base réglé), qui est essentiellement une collection de "stations" (STAs), deux ou plus, sans fil, qui peuvent communiquer entre eux. Si les stations communiquent directement avec d'autres stations dans leur secteur de couverture (communication de pair-à-pair), le BSS est désigné sous le nom d'un IBSS (BSS indépendant) ou est dit travailler en mode *ad hoc*. Autrement, si tous les nœuds sans fil communiquent entre eux par l'intermédiaire d'un point d'accès fixe (AP), le BSS est dit en mode *infrastructure*. L'AP peut être considéré comme un pont entre les réseaux câblés et le monde sans fil. L'ensemble prolongé de service (ESS) se compose d'une série de BSSs reliées ensemble au moyen d'un système de distribution (DS). Ceci est montré dans la figure A.1.

MANET - L'objectif du groupe MANET [man] est de normaliser la fonctionnalité du protocole de routage d'IP, adaptée à des topologies statiques et dynamiques. Les approches sont prévues pour être relativement légères et de nature approprié au matériel diversifié et aux environnements sans fil, et les scénarios adressés sont ceux où les MANETs sont déployés aux bords d'une infrastructure d'IP. Les infrastructures hybrides maillées (par exemple, un mélange des nœuds fixes et mobiles) sont également censées être soutenues par des spécifications MANET et des dispositifs de maintenance. Les conditions nécessaires au routage sécurisé, et les problèmes de routage sont également adressés. Le groupe de travail de MANET examine deux catégories des protocoles de routage.

1. Protocoles réactifs : les protocoles réactifs de routage permettent le routage dynamique et de multi-saut entre un ensemble de nœuds participants. Ils exigent d'échanger des messages seulement quand un nouvel (ou non-existant) itinéraire entre une source et un nœud destinaire doit être découvert ou quand un itinéraire existant échoue. Les protocoles DSR [dsr], DYMO [dym] sont les

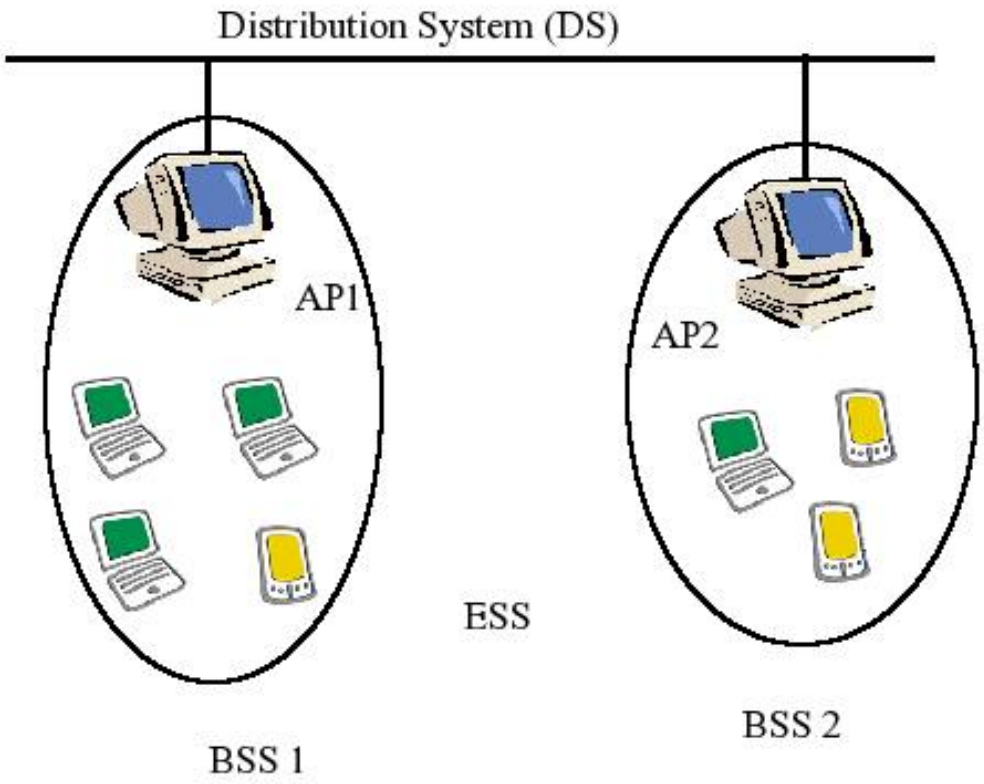


Figure A.1: IEEE 802.11 Network

802.2			Data−Link
802.11 MAC (CSMA/CA)			
FHSS	DSSS	InfraRed	Physical

Figure A.2: The IEEE 802.11 Stack

deux Internet-drafts dans cette catégorie.

2. Protocoles proactifs : en revanche, les protocoles proactifs de routage exigent l'échange périodique d'information de routage pour les itinéraires libres de boucles et corrects. Le protocole OLSR [ols] est une Internet draft de cette catégorie.

Sécurité dans réseaux ad hoc

Nous présentons les nouveaux problèmes qui sont dûs à la nature différente de ces réseaux. Nous présentons en détail deux problèmes particulièrs; celui de l'établissement de clefs et delui du routage sécurisé, qui sont fondamentaux pour l'installation d'autres services de sécurité dans les réseaux ad hoc, et qui sont les domaines d'étude principaux dans cette thèse. Contrairement aux réseaux câblés, les réseaux ad hoc n'ont aucune sécurité physique de bureaux ou de bâtiments confinés avec l'isolement du medium de communication sous forme de fils. Ainsi les moyens cryptographiques de la sécurité jouent un rôle important. Et assurer une communication sûre dans des réseaux qui en peuvent être formés en endroit ouvert (forêts, champs de bataille y compris), et qui emploient un medium sans fil n'est en aucun cas facile. Nous énumérons au-dessous des problèmes de base qui peuvent être identifiés pour un réseau ad hoc [HBC01, SA99, Sta01].

Contrôle d'accés et authentification : contrôler qui peut ou ne peut pas faire partie du réseau n'est pas une problème insignifiant. Tandis que dans les réseaux câblés traditionnels, l'endroit bloqué du point d'accès de réseau impose quelques critères d'admission, quelque chose de semblable manque dans les réseaux ad hoc sans fil. Non seulement n'importe quel nœud avec une antenne appropriée peut recevoir des messages étant transmiss au-dessus d'un réseau sans fil voisin, mais aussi il est également tout à fait simple d'introduire des messages dans le réseau. Ainsi en l'absence de procédé de contrôle d'admission, n'importe quel nœud malveillant peut désorganiser le fonctionnement du réseau. Des solutions non-cryptographiques au problème du contrôle d'accès sont la plupart du temps basées sur l'évaluation et la gestion de confiance. Essentiellement l'idée ici est que les nœuds calculent un niveau de confiance

pour d'autres nœuds, basé sur l'évidence liée aux interactions précédentes avec eux, ou aux niveaux de la confiance qu'ont d'autres nœuds pour ce nœud. Les exemples des travaux dans cet domaine incluent [BT04, BEG02]. Les solutions cryptographiques se fondent essentiellement sur l'authentification des nœuds (employant la plupart du temps la cryptographie à clé publique) et ensuite des politiques pour définir les conditiones d'accès au réseau. Comme l'établissement d'une infrastructure de clé publique normale (PKI) est souvent considérée infaisable dans de tels réseaux, des approches modifiées comprenant une PKI distribuée cmme [YK03, YK02], ainsi bien qu'une approche comme PGP [CBH03] sont suggérées. Pour les protocoles proposés dans cette thèse, nous avons besoin des mécanismes d'authentification à clé publique, et nous supposons qu'une PKI distribuée comme [CBH03] existe.

L'établissement et la gestion de clefs : les clefs sont l'ingrédient de base dans beaucoup de protocoles cryptographiques, et les protocoles d'établissement et de gestion de clefs fournissent un mécanisme pour établir des clefs dans un réseau, et pour les contrôler sur une période prolongée. C'est l'un de sujets principaux de cette thèse. On présente ce sujet en détail plus tard.

Sécurisation du routage : dans les réseaux ad hoc multi-saut, chaque nœud doit contribuer à faciliter le processus du routage, car il n'y a aucun nœud dédiéé pour ce faire. Ceci crée également le problème que quelques nœuds peuvent perturber le processus du routage. La rupture peut aussi bien être provoquée par deux nœuds légitimes (des nœuds authentifiés et autorisés) du réseau, que par des nœuds illégitimes (nœuds cachés ou adversaires). Le but d'un nœud malveillant peut être de causer la perte des paquets vers une destination particulière, ou de conduire certains paquets à transiter par lui ou le nœud voisin, entre d'autres. Il semble impossible d'empêcher toutes sortes d'attaques dans de tels réseaux coopératifs. Mais le but principal de sécurisation du routage devrait être de réduire au minimum et localiser l'effet de tels des attaques, et d'être capable de revenir à la normale une fois que les nœuds disruptifs sont identifiés et/ou enlevés. Nous élaborons les problèmes liés aux routage plus tard.

Etablissement et gestion de clefs

L'objectif principale de l'établissement de clef est de s'assurer que tous les nœuds dans le réseau ont le *matériel de cles* nécessaire pour certains des services de sécurité mentionnés ci-dessus. Le*matériel de cle* peut inclure des clefs publiques ou privées, des valeurs d'initialisation, et n'importe quels autres paramètres publics. Le processus de l'établissement de clé n'est pas une affaire à un coup, les clefs devant être remplacées par exemple : a) si des nouveaux nœuds arrivent dans le réseau, b) quand quelques nœuds partent du réseau, c) quand quelques nœuds sont compromis, ou d) quand la clé de vie expire. Ainsi nous avons besoin d'un service dynamique d'établissement de clef qui doit continuer à fournir les clefs nécessaires pendant la vie du réseau. Chaque clef a un but, une période de la validité (appelée sa *duree de vie*), et un ensemble de propriété (entités possédant la clef). Ainsi dans certains scénarios, nous pourrions avoir une clef simple pour tous les nœuds dans le réseau pendant la vie du réseau, alors que dans d'autres scénarios nous pourrions avoir une clef unique pour chacun ensemble bien défini de nœuds, pendant une période bien définie. Une clef employée par un ensemble bien défini de nœuds (nous nous référerons à cet ensemble en tant que groupe), pendant une période donnée, est appelée clef de *session* (ou clef de groupe en cette dissertation), car typiquement elle est employée pour fixer une session de communication (pendant une certaine durée) entre les nœuds. En outre un nœud particulier peut être un membre de multiples (disjoints ou se recouvrant) groupes (ou sessions), à un moment donné, et avoir ainsi des clefs multiples de session. Une clef qui dure typiquement pendant toute la vie d'un nœud est désignée souvent sous le nom de sa clef à *long terme*. Pas mal des protocoles d'établissement de clef existent dans la littérature. Nous donnons ci-dessous quelques définitions formelles et discutons les diverses classes des protocoles d'établissement de clef.

Les protocoles d'établissement de clefs peuvent être grossièrement classifiés en deux catégories, à savoir : le *transport de clé* et l'*accord de clé*, et peuvent utiliser des techniques symétriques aussi bien que des techniques cryptographiques asymétriques. Nous donnons des définitions formelles de l'accord principal de transport et de clef et examinons de tels protocoles.

Definition A.1 *Transport de clef : un mécanisme d'établissement de clé où une*

partie crée une valeur secrète, et ensuite la transfère à d'autres.

Definition A.2 *Accord de clé : un mécanisme d'établissement de clé où une valeur secrète commune est dérivée à partir de contributions de chacun, et d'une manière telle que personne ne peut pré-déterminer la clef avant la fin de l'exécution du protocole.*

Definition A.3 *L'établissement authentifié de clef : un protocole principal d'établissement de clé qui fournit la propriété de l'authentification de clé, par laquelle chaque partie est assurée qu'aucune partie, hormis les parties de confiance identifiées, ne peut accéder à la clef secrète partagée.*

Definition A.4 *Forward secrecy : on dit qu'un protocole fournit la propriété de* forward secrecy *si la compromission de la clef à long terme d'un des particpants ne compromet pas les clefs des sessions précédentes.*

Transport de clé en utilisant la cryptographie symétrique : l'idée essentielle derrière de tels protocoles est qu'un tiers de confiance, ou un des membres du groupe, choisit (ou obtient) la clef, et la transfère à tous les autres en utilisant le chiffrement symétrique. Ceci suppose qu'une valeur secrète à long terme est a priori partagée entre l'expéditeur et tous les destinataires. Les exemples de cette classe des protocoles (pour deux parties seulement) incluent le protocole célèbre de Kerberos [MNSS87], le protocole de Needham et Schroeder [NS78], et celui d'Otway et Rees [OR87] entre autres. De tels protocoles utilisent la cryptographie symétrique et sont ainsi efficaces en général. Le problème principal avec de tels protocoles, pour être utilisés dans les réseaux ad hoc, est la nécessité d'avoir une clef secrète partagée a priori entre l'expéditeur et tous les autres. L'établissement de $\frac{n(n-1)}{2}$ clés a priori dans un groupe de la taille n est tout à fait inefficace. Si un secret à long terme simple est partagé entre toutes les parties, il mène à la perte de *forward secrecy.*

Transport de clé en utilisant la cryptographie asymétrique : ici, une partie choisit une clef symétrique et l'envoie à d'autres en utilisant le chiffrement asymétrique.

Si l'authentification est considérée comme importante, des signatures peuvent être aussi incluses dans le protocole. En termes d'efficacite, ces protocoles sont plus coûteux que des protocoles symétriques de transport. Et, de nouveau, la révélation de la clef privée de n'importe quelle partie, compromet toutes les clefs envoyées cette pratie précédemment. Donc cela ne fournit pas la propriété de forward secrecy. Les exemples incluent [BY93, NS78].

L'accord de clé employant la cryptographie aymétrique [1] : le protocole fondamental de Diffie-Hellman est un exemple de cette classe de protocole. Beaucoup des protocoles bipartites, ainsi que des protocoles à n parties ont été proposés basée sur ce protocole, et incluent STS [vO93], MTI [MTI86]. La plupart des protocoles dans cette catégorie fournissent la propriété de forward secrecy. Nous discutons dans les détails beaucoup de protocoles existants aussi bien qu'un nouveau protocole principal d'accord de clé pour n parties en Section "".

Sécurité du Routage

En raison du manque d'infrastructure fixe dédiée dans les réseaux ad hoc, tous les nœuds doivent coopérer pour expédier leurs messages. Ceci rend le processus de routage plus enclin aux attaques, car n'importe lequel des nœuds peut se conduire mal et faire effondrer le routage. En raison de la mobilité, il devient aussi plus dur de trouver et d'isoler le nœud se conduisant mal. Nous définissons avec précision les attaques identifiées sur le routage [HP04, MM03].

Cache poisoning : un nœud malveillant pourrait annoncer une métrique zéro pour tous les nœuds destinaires, ou annoncer des liens inexistants vers certaines destinations. Ceci ferait que tous les autres nœuds expédient des paquets pour toutes ces destinations, en passant par le nœud de l'attaquant, et ce dernier peut ensuite laisser tomber ces paquets, perturbant ainsi le cheminement dans le réseau.

Black hole attack: une attaque plus passive est juste de laisser tomber certains messages de routage qu'un nœud reçoit. La chute des paquets pourrait être faite

[1] Le seul exemple connu d'accord de cle employant le cryptographie symétrique est [Blo84] mais il exige d'un serveur central pour la pré-distribution des clefs.

d'une manière que les soupçons au sujet d'un nœud malveillant ne sont pas augmentés. Ceci diminue la quantité d'information de routage disponible au réseau pour prendre les décisions. Il est difficile de détecter une telle attaque, car elle peut être facilement confondue avec un défaut de fonctionnement d'un lien. Ainsi la notion de robustesse du protocole est plus appropriée pour contrer de telles attaques, plutôt que des mécanismes cryptographiques.

Message tampering (modification de messages) : un nœud a pu modifier les messages du routage d'autres nœuds avant de les expédier. C'est tout à fait facile de le faire s'il n'y a aucun contrôle d'authentification des messages.

Replay attacks (attaques de rejeu) : un nœud pourrait propager les vieux messages de routage (qui ne reflètent pas la topologie courante) dans le réseau pour affecter les itinéraires. Cette attaque fonctionne même s'il y a des techniques d'authentification de message employé par les nœuds.

Wormhole attacks : si un nœud malveillant X est dans la zone de communication de deux nœuds qui ne sont pas eux-mêmes pas dans une même zone commune, X peut simplement expédier les messages d'un nœuds à autre (agir comme un relais), et les deux nœuds victimes considèrent qu'ils sont dans la même zone de communication. Ensuite le nœud X peut simplement arrêter ce relais des messages. L'attaque devient plus significative si cette attaque est portée entre deux nœuds de bord de deux sous-réseaux différents.

Rushing attacks : dans des protocoles réactifs de routage, si un nœud peut parvenir à expédier des demandes d'itinéraire (RREQs) au nœud destinaire beaucoup plus tôt que les RREQs par d'autres itinéraires (souvent avec l'aide d'un autre nœud malveillant), il a une probabilité de forcer des itinéraires à passer par lui.

La description ci-dessus des attaques indique clairement qu'il y a une gamme énorme d'attaques possibles sur des protocoles de routage dans les réseaux ad hoc. Et souvent il est difficile de distinguer une attaque d'un défaut de fonctionnement (dû à échecs de lien, à congestion). Ainsi le but des protocoles sûr de routage devrait être d'avoir des mesures permettant d'éviter les attaques qui affectent sévèrement le réseau entier,

et d'être assez robuste pour faire face aux attaques localisées et spécialisées par des nœuds malveillants.

Cryptographie dans les réseaux en practique

La plupart des protocoles cryptographiques formulent des hypothèses au sujet de l'environnement de l'exécution du protocole. En particulier, au sujet du medium de communication, les hypothèses suivantes sont fréquemment employées :

1. Fiabiblité : les messages envoyés par le participant A au participant B sont reçus intacts et dans l'ordre.
2. Synchronisation : comme la plupart des protocoles spécifient des étapes chronologiques souvent déclenchées par la réception d'un des messages du protocole, il y a une hypothèse au sujet de la capacité des participants à réaliser une communication synchronisée.
3. Émission (broadcast) : les protocoles se fondant sur un milieu d'émission tels qu'un message peut être à direction de tous les nœuds, et effectivement reçu par ceux-ci..

Il est clair que dans les réseaux ad hoc, on ne peut pas faire de telles hypothèses car les messages peuvent être perdus, ou arriver avec différentes délais pour différents récepteurs. Ainsi le comportement de protocoles cryptographqiues faisant de telles hypothèses dans les réseaux ad hoc devient imprévisible. Tandis que quelques protocoles peuvent simplement échouer, d'autres peuvent voir leurs objectifs de sécurité compromis. Dans les sections suivantes, nous proposons de nouveaux protocoles cryptographiques pour les réseaux ad hoc. La présentation initiale des protocoles est faite suivant la présentation standard des protocoles cryptographiques, et les hypothèses mentionnées ci-dessus sont faites. Mais ensuite, nous présentons aussi une section sur l'exécution de ces protocoles dans des environnements réalistes, où nous relâchons ces hypothèses, ce qui nous conduira à de légères modifications des protocoles. Le modèle de sécurité et les preuves de ces protocoles modifiés doivent être examinés de nouveau dans ce modèle. Ceci conclut notre introduction aux titres dans les réseaux ad hoc.

Dans la prochaine section, nous explorons en détail le domaine de l'établissement de cle dans les réseaux ad hoc.

Établissement de clefs dans les réseaux Ad hoc (Chapitre 3)

L'établissement de cle est un service cryptographique de base, requis avant que d'autres services de sécurité puisse être déployés dans le réseau. La charge d'établir et de maintenir des clefs cryptographiques dans un réseau ad hoc est compliquée par l'absence d'une autorité de confiance permanente, et la composition changeant rapidement du réseau. Ainsi il y a aucun nœud qui ne peut se charger de fournir des clefs á d'autres nœuds. Et comme la composition du réseau change souvent, c'est souvent une propriété souhaitable que les clefs soient changées en conséquence, pour s'assurer que seulement les nœuds actuellement dans le réseau puissent accéder aux données étant échangées. Les clefs peuvent également être exigées d'être changées si un certain nœud est compromis par un adversaire, qui a ainsi un accès illégitime á certaines des clefs étant employées dans le réseau. Ainsi dans ce qui suit sont présentées les conditions souhaitables d'un mécanisme d'établissement de clé pour les réseaux ad hoc :

- **Décentralisé** : le service d'établissement de clé ne devrait pas se fonder sur un certain nœud de confiance central fixe, supposé être présent tout le temps. Comme la connectivité des nœuds dans un réseau ad hoc peut être divisée tout à fait abruptement, le service entier peut être perturbé s'il se fonde sur un nœud simple. Un service fiable d'établissement doit ainsi être réparti sur plusieurs nœuds. En raison du rapport de pair-à-pair des nœuds dans un réseau ad hoc, les clefs, de préférence, ne devrait pas être décidées par une entité simple, mais plutôt, collectivement par les nœuds.

- **Dynamicité** : le service d'établissement de clé devrait pouvoir gérer le départ

des nœuds existants et l'arrivée de nouveaux nœuds dans le réseau, sans exiger une réexécution complète du protocole d'établissement de cle. Souvent il est désirable que les nouveaux nœuds entrants ne puissent pas accéder à l'information échangée dans le passé, et les nœuds qui partent ne doivent pas pouvoir accéder à l'information concernant des sessions ultérieures. Ainsi il devient essentiel que les clefs soient changées pour fournir la propriété de l'indépendance de cle à travers différentes sessions dans le réseau et dans le temps.

- **Efficacité** : le service devrait être assez léger de sorte que les nœuds dépensent seulement une fraction de leur temps en établissement des clés.

L'etat d'art Comme déja discuté, les protocoles d'établissement de clef peuvent être largement classifiés dans trois catégories. Nous étudions ici la praticabilité de ces schémas dans les réseaux ad hoc et présentons la littérature existante appropriée.

A) Transport de clé en utilisant la cryptographie symétrique : comme les réseaux ad hoc manquent de la présence d'une autorité en ligne pour distribuer les clefs, la plupart des protocoles traditionnels de transport de clé employant la cryptographie symétrique ne sont pas bien adaptés pour de tels réseaux. Récemment, des propositions qui emploient seulement la cryptographie symétrique en l'absence d'autorité permanente (une autorité de confiance est exigée seulement avant que le déploiement) ont été faites, et emploient l'idée de la pré-distribution de clé [CPS03, DDH$^+$05, EG02]. De tels protocoles exigent qu'à chaque nœud dans le réseau soit assigné un sous-ensemble de clefs d'un pool commun de clefs, au préalable, avant que les nœuds soient déployés réellement dans le réseau. La taille du pool commun et du sous-ensemble de clefs qu'un nœud partage avec chaque nœud doit être choisie afin d'assurer une probabilité forte qu'un nœud trouve une clef commune avec d'autres nœuds. De telles solutions sont plutôt applicables aux réseaux des capteurs où tous les nœuds sont déployés par une autorité commune. Dans les réseaux ad hoc les mécanismes principaux de pré-distribution paraissent beaucoup moins faisables.

B) Transport de clé en utilisant la cryptographie asymétrique : les mécanismes traditionnels du transport de clé employant la cryptographie asymétrique sont peu convenables car ils exigent une autorité de confiance en ligne. Dans les réseaux ad hoc, le transport de clef employant la cryptographie asymétrique peut être réalisé en choisissant un membre comme serveur principal pendant une certaine période (session), qui choisit une clef pour le groupe et et l'envoie à tous les nœuds, en la chiffrant en utilisant leurs clefs publiques respectives. Mais le fait qu'un des nœuds choisit la clef pour le groupe entier peut ne pas être acceptable dans la plupart des réseaux ad hoc qui sont des réseaux pair-à-pair, avec seulement une confiance partielle entre les membres. En outre, la révélation de la clef secrète d'un certain nœud compromet toutes les sessions où ce nœud particulier participait.

C) L'accord de clé employant la cryptographie aymétrique : les protocoles d'accord de clé pour des groupes permettent aux nœuds de coopérer, et d'arriver à une clef commune de groupe sans besoin de serveurs de confiance dédiés. Ces protocoles peuvent également s'adapter à la dynamique du réseau en fournissant des moyens efficaces de changement la clef de groupe au fur et à mesure que cela est requis. Ainsi ces protocoles semblent répondre à la plupart des besoins d'un service d'établissement de clé pour les réseaux ad hoc. Nous analysons et étudions l'adéquation des protocoles existants d'accord de clef de groupe, pour les réseaux ad hoc.

La plupart des protocoles d'accord de clef de groupe sont dérivés du protocole bipartite d'échange de clef de Diffie-Hellman. Les protocoles de GKA, non basés sur Diffie-Hellman, sont peu nombreux et incluent [PL00, TT00, BN03]. Les deux protocoles de Li [PL00] et Boyd [BN03] ne fournissent pas la propriété de *forward secrecy* tandis que le protocole de Tzeng [TT00] est très gourmand en ressources et enclin à certaines attaques [BN03]. La propriété de forward secrecy est une propriété très souhaitable pour les protocoles d'établissement de clé dans les réseaux ad hoc, car quelques nœuds peuvent être facilement compromis à cause de la faible sécurité

physique des nœuds. Ainsi il est essentiel que la compromission d'un nœud ne compromette pas toutes les clefs des session suivantes. Nous récapitulons et comparons des protocoles existants du GKA basés sur des protocoles de Diffie-Hellman dans le tableau A.1. Nous comparons essentiellement des versions sans authentification, car la plupart réalisent l'authentification en employant des signatures numériques et ont ainsi les coûts similaires en ce qui concerne l'authentification. Nous comparons l'efficacité de ces protocoles basés sur les paramètres suivants.

- **Nombre de rounds synchrones** : dans un round synchrone simple, de multiple messages indépendants peuvent être échangés dans le réseau. Bien qu'on ait montré dans [BW98] (voir également le tableau A.2), que au moins n messages sont exigés pour que soit mise en place une clé contributive entre n participants, le nombre de rounds synchrones dans lesquels ces messages peuvent être échangés peut être inférieur si les messages sont indépendants. Le temps requis pour executer un protocole de GKA, efficace en nombres de rounds, peut être beaucoup moins que dans d'autres protocoles de GKA qui ont le même nombre de messages totaux mais de plus de rounds. C'est parce que les nœuds passent moins de temps à attendre d'autres messages avant d'envoyer leurs propres messages.

- **Nombres de messages** : c'est le nombre de messages (unicast ou broadcast) échangés dans le réseau pour dériver la clef de groupe. Pour les réseaux ad hoc multiple-hop, la distinction entre l'unicast et les messages à diffusion générale (broadcast) est importante, car les messages de diffusion peut consommer beaucoup plus d'énergie que les messages unicast.

- **Nombres d'exponentiations** : tous les protocoles de GKA basés sur Diffie-Hellman exigent qu'un certain nombre d'exponentiations modulaires soient exécutées par chaque participant. Une exponentiation modulaire est une opération intensive en termes d'effort de calcul, comparée à d'autres opérations, et les compter donne ainsi une bonne indication du coût de calcul pour chaque nœud.

	Rounds	Messages	Broadcasts	Expo par U_i
ITW [ITW82]	$m-1$	$m(m-1)$	0	m
GDH.1 [STW96]	$2(m-1)$	$2(m-1)$	0	$i+1$
GDH.2 [STW96, BCP02]	m	$m-1$	1	$i+1$
GDH.3 [STW96]	$m+1$	$2m-3$	2	3
Perrig [Per99]	$\log_2 m$	m	$m-2$	$\log_2 m+1$
Dutta [DB05]	$\log_3 m$	m	m	$\log_3 m$
Octopus [BW98]	4	$3m-4$	0	4
BDB [BD94, KY03]	2	$2m$	m	3
BCEP [BCEP03]	2	$2m$	0	$2^{\dagger}$
Catalano [BC04]	2	$2m$	0	$m+1$
NKYW [NLKW04]	2	m	1	$2^{\ddagger}$
KLL [KSML04]	2	$2m$	$2m$	3
STR [SSDW88, KPT04]	2	m	1	i^{*}
Notre [ABIS05]	2	m	1	2^{**}

$\dagger$: m exponentiations pour la station de base.
$\ddagger$: $m+1$ exponentiations et $m-1$ calcul d'inverse pourela nœud parent.
$*$: $2m$ exponentiations pour la nœud sponsor.
$**$: m exponentiations pour le chef.

Table A.1: Comparaison de protocoles de GKA, avec m participants

Les coûts de communication demeurent toujours le facteur critique pour choisir des protocoles eficace en énergie pour la plupart des réseaux ad hoc. Une exponentiation modulaire peut être exécutée en quelques dizaines de millisecondes sur la plupart des ordinateurs portables, tandis que la propagation de message dans les réseaux ad hoc de multi-hop peut être facilement de l'ordre de quelques secondes, et à des implications sur la consommation d'énergie des nœuds dans le réseau. Comme vu dans le tableau A.1, quelques protocoles existants de GKA exigent des rounds de $O(n)$ de communication pour n participants au protocole. De tels protocoles ne passent pas bien à l'échelle dans les réseaux ad hoc. Même les protocoles de GKA qui ont nombre de rounds de $O(\log m)$ peuvent tout à fait coûteux pour les réseaux ad hoc. Seulement des protocoles avec un nombre constant de rounds sont appropriés pour les réseaux ad hoc. Dans ce qui suit, nous présentons ces protocoles constants du round GKA brièvement.

Ainsi, parmi les protocoles à nombre constants de rounds, Octopus [BW98], BDB [KY03] et KLL [KSML04] exigent un arrangement spécial des participants. Ceci a la conséquence que les messages envoyés par un participant dépendantes de celui des autres. En ce cas, l'échec d'un nœud peut souvent arrêter le protocole. Ainsi de tels protocoles ne sont pas assez robustes pour s'adapter bien au dynamisme des réseaux ad hoc. Le protocole BCEP [BCEP03] ne fournit pas la propriété de forward secrecy, car si le secret à long terme de la station de base est rélévé, les clés de session précédentes sont compromises. Le protocole de Catalano [BC04] exige un calcul d'ordre $O(m)$ pour chaque participant. Un autre inconvénient est que si le message d'un participant est perdu dans le premier round, le protocole st arrêté, car le schéma de secret partagé employé implique que toutes les contributions de m sont exigées pour calculer la clef. Le protocole NKYW [NLKW04] bien qu'efficace, ne fournit aucun procédé pour les changements de clé quand la composition du groupe change. Le protocole STR [SSDW88] reste un peu cher avec jusqu'à $m - i$ exponentations pour le participant M_i. Le protocole manque d'une preuve de sécurité contre les adversaires actifs. Cette incapacité des protocoles existants de GKA d'être efficacement mis en application dans les réseaux ad hoc était la motivation pour notre recherche dans ce secteur. Un protocole efficace et simple, facile à mettre en application dans les réseaux ad hoc était le but. À cet effet, nous présentons un nouveau protocole d'accord de clef de groupe dans les sections suivantes avec une analyse de sécurité et des détails d'exécution. Le protocole n'exige pas des participants d'être arrangés suivant un certain ordre fixé avant que le protocole commence. En fait le protocole peut être lancé par n'importe quel participant possible (à qui nous nous référons en tant que chef de *group*) et ensuite, la formation de groupe et le calcul de la clé de groupe peut être faits simultanément. En outre, la version de protocole sans authentification rassemblements de protocole rencontre la limite inférieure (voir le Tableau A.2) pour le nombre de messages et dépasse la limite pour les rounds synchrones par un.

Un nouveau protocole de GKA - Nous présentons un nouveau protocole d'accord de clef de groupe dérivé d'une version modifiée de l'échange de clef de Diffie-Hellman de deux parties. Dans ce qui suit, nous employons les termes *session* et *group* par

	Messages	Rounds simple	Rounds Synchrones
Sans diffusion	$2(m-1)$	$\log_2 m$	-
Avec diffusions	m	$\log_2 m$	1

Table A.2: Bornes Inferieures de GKA [BW98]

intermittence, mais tous les deux se réfèrent à un certain sous-ensemble de nœuds[2] dans le réseau, exécutant le protocole de GKA pour convenir d'une clef secrète. Nous employons le term *chef de group* pour dénoter le nœud qui débute le protocole de GKA. Comme notre protocole permet à n'importe quel nœud de commencer le protocole, n'importe quel nœud dans le réseau peut être le chef de groupe pour une session particulière. Notre protocole de GKA n'impose aucune contrainte quant au choix du chef de groupe, qui pourrait être aussi bien choisi aléatoirement, ou, par les contraintes de l'application sous-jacente. Plus de détails sur l'élection du chef de groupe peuvent être trouvés dans la section A. Le protocole est conçu pour les groupes dynamiques et à ainsi trois protocoles secondaires notamment:

a) **Initial Key Agreement (IKA)** - pour qu'un groupe dérive un nouveau clef de groupe en utilisant chaque contribution.

b) **Merge** - pour qu'un groupe dérive une nouvelle clef de groupe quand de nouveaux membres joignent un groupe existant.

c) **Partition** - pour qu'un groupe dérive une nouvelle clef de groupe quand quelques membres laissent le groupe existant.

Protocol sans authentification

Notation :

G : un sous-groupe (d'ordre premier q, de générateur g) d'un groupe.

$\mathcal{M}$: l'ensemble des indices des participants à une session.

M_i : le i-ième participant parmi les n participants à une session.

[2]Le terme "nœud" lui-même sont synonymes des termes "participant" et "membre" dans cette dissertation et elles sont employés par intermittence selon le contexte.

M_l : le chef de groupe ($l \in \{1, ...n\}$) dans une session.
r_i : un nombre aléatoire (dans $[1, q-1]$) engendré par le participant M_i. Aussi appelé le *secret* de M_i.
g^{r_i} : le *secret aveugle* de M_i (mod q).
$g^{r_i r_l}$: la *réponse aveugle* pour M_i (mod q) calculée par M_l.
$\mathcal{J}$: l'ensemble des indices d'un groupe de participants désirant se joindre au groupe existant.
$\mathcal{D}$: l'ensemble des indices d'un groupe de participants désirant quitter le groupe existant.
$x \leftarrow y$: x prend la valeur y.
$x \stackrel{r}{\leftarrow} \mathcal{S}$: x est tiré aléatoirement dans S, suivant la distribution uniforme.
$M_i \longrightarrow M_j : \{msg\}$: M_i envoie le message msg au participant M_j.
$M_i \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg\}$: M_i broadcaste le message msg à tous les participants dans le groupe indexé par $\mathcal{M}$.

Étapes du protocole :
Round 1 : chaque M_i répond à la demande initiale ($INIT$) de M_l, le chef courant de groupe (qui est aléatoirement choisi), avec son secret aveugle g^{r_i}.

Round 2 : M_l rassemble tous les secrets aveugles reçus, élève chacun d'eux à la puissance son secret (r_l) et les émet en diffusion (broadcast) avec les contributions originales au groupe, *i.e.* il envoie $\{g^{r_i}, g^{r_i r_l}\}$ pour tout $i \in \mathcal{M} \setminus \{l\}$.

Calcul de clé : chaque M_i vérifie si sa contribution est incluse correctement et enlève alors son secret r_i de $g^{r_i r_l}$ pour obtenir g^{r_l}. La clef de groupe est

$$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l (1 + \sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}.$$

Note :
1) les contributions originales g^{r_i} sont incluses dans le dernier message car elles sont exigées pour le calcul de clef en cas de modifications de groupe (voir ci-dessous);
2) bien que le $\Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l}$ soit publiquement connu, il est inclus dans le calcul principal, pour dériver une clef qui dépende bien de chaque contribution.

Round 0
$l \xleftarrow{r} \mathcal{M}, M_l \xrightarrow{B} \mathcal{M} : \{INIT\}$
Round 1
$\forall i \in \mathcal{M} \setminus \{l\}, r_i \xleftarrow{r} [1, q-1], M_i \longrightarrow M_l : \{g^{r_i}\}$
Round 2
$r_l \xleftarrow{r} [1, q-1], M_l \xrightarrow{B} \mathcal{M} : \{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}$
$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table A.3: Agrément initial de clé : IKA

Le protocole est formellement défini dans le Tableau A.3. Nous voyons maintenant comment ce protocole peut être employé pour dériver des procédures d'IKA, de Join/Merge et de Delete/Partition pour les réseaux ad hoc.

Initial Key Agreement - Quand un groupe de participants M_i, $i \in [1, n]$ souhaite calculer une clef de groupe, chaque participant envoie son secret aveugle g^{r_i} à un chef de groupe M_l ($1 \leq l \leq n$) aléatoirement choisi. Un message simple (Round 2 dans Tableau A.3) du chef de groupe, M_l, (en utilisant des contributions des autres participants) permet à chaque membre de calculer la clef de groupe. Un exemple est fourni ci-dessous.

Supposons que M_1 initie le protocole de GKA et 3 membres envoient leurs secrets aveugles, g^{r_2}, g^{r_3} et g^{r_4} respectivement. Puis le chef de groupe, M_1, annonce le message suivant : $\{g^{r_2}, g^{r_3}, g^{r_4}, (g^{r_2})^{r_1}, (g^{r_3})^{r_1}, (g^{r_4})^{r_1}\}$. À la réception de ce message, chaque membre peut dériver g^{r_1} en utilisant son secret respectif. Ainsi la clef $g^{r_1(1+r_2+r_3+r_4)}$ peut être calculée.

Join/Merge - Join est tout à fait semblable à IKA. Chaque participant entrant, M_i ($i \in \mathcal{J}$), émet une demande de $JOIN$ avec son secret aveugle, g^{r_i}. Le chef de groupe existant (M_l) met à jour la composition en groupe et choisit un nouveau secret aléatoire, r_l, et envoie tous les secrets aveugles (y compris deux des membres se joignants) au nouveau chef de groupe, $M_{l'}$, (qui peut être choisi aléatoirement). Le nouveau chef de groupe annonce un message semblable au message round de 2 dans IKA, *i.e.* tous les secrets aveugles et les reponses aveugles. Il vaut la peine de noter

Round 0
$\forall i \in \mathcal{J}, r_i \overset{r}{\leftarrow} [1, q-1], M_i \overset{B}{\longrightarrow} \mathcal{M} : \{JOIN, g^{r_i}\}$
Round 1
$r_l \overset{r}{\leftarrow} [1, q-1], \mathcal{M} = \mathcal{M} \cup \mathcal{J}, l' \overset{r}{\leftarrow} \mathcal{M}, M_l \longrightarrow M_{l'} : \{g^{r_i}\}_{i \in \mathcal{M} \setminus \{l'\}}$
Round 2
$l \leftarrow l', r_l \overset{r}{\leftarrow} [1, q-1], M_l \overset{B}{\longrightarrow} \mathcal{M} : \{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}$
$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table A.4: Protocole Join/Merge

que le nouveau chef de groupe écarte le secret qu'il a employé pendant la demande de $JOIN$ (ou le secret de la dernière session) et produit un nouveau secret aléatoire pour le message à diffusion générale. Voir le tableau A.4 pour des spécifications formelles et ci-dessous pour un exemple.

Supposons que les nouveaux membres, M_9 et M_{10} joignent le groupe de M_1, M_2, M_3 et M_4 avec leurs contributions g^{r_9} et $g^{r_{10}}$ respectivement. Alors le chef de groupe précédent (M_1) change son secret à r_1^* et envoie $g^{r_1^*}$, g^{r_2}, g^{r_3}, g^{r_4}, g^{r_9} à M_{10} (qui sera le nouveau chef de groupe). M_{10} produit un nouveau secret r_{10}^* et annonce le message suivant au groupe : $\{g^{r_1^*}, g^{r_2}, g^{r_3}, g^{r_4}, g^{r_9}, g^{r_{10}^* r_1^*}, g^{r_{10}^* r_2}, g^{r_{10}^* r_3}, g^{r_{10}^* r_4}, g^{r_{10}^* r_9}\}$. La clef est $g^{r_{10}^*(1+r_1^*+r_2+r_3+r_4+r_9)}$.

Delete/Partition - Quand les membres laissent le groupe, un nouveau chef de groupe est aléatoirement choisi parmi les membres restants, et il change sa contribution secrète et envoie un message au groupe, omettant la contribution du membre partant. Voir le tableau A.5 ci-dessous pour un exemple.

Supposons qu'un membre, M_2, quitte le groupe de M_1, M_2, M_3, M_4, M_9 et M_{10}. Le chef précédent, M_{10} change son secret à $r_{10}^{''}$ et émet $\{g^{r_1^*}, g^{r_3}, g^{r_4}, g^{r_9}, (g^{r_1^*})^{r_{10}^{''}}, (g^{r_3})^{r_{10}^{''}}, (g^{r_4})^{r_{10}^{''}}, (g^{r_9})^{r_{10}^{''}}\}$. Le clef est $g^{r_{10}^{''}(1+r_1^*+r_3+r_4+r_9)}$.

Dans le protocole présenté, les clefs de différentes sessions sont maintenues indépendantes en exigeant au moins d'un membre (habituellement le chef de groupe) de changer sa contribution secrète entre chaque session. En outre le rôle du chef de groupe peut

Round 0
$\forall i \in \mathcal{D}, M_i \xrightarrow{B} \mathcal{M} : \{DELETE\}$
Round 1
$\mathcal{M} = \mathcal{M} \setminus \mathcal{D}, l \xleftarrow{r} \mathcal{M}, r_l \xleftarrow{r} [1, q-1],\ M_l \xrightarrow{B} \mathcal{M} : \{g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}$
$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table A.5: Protocole Delete/Partition

être pris par un nœud différent à travers différentes sessions.

Protocole avec authentification - Le protocole présenté ci-dessus ne présente pas l'authentification des messages et est sûr contre les adversaires passifs seulement. Nous montrons maintenant comment convertir ce protocole en un protocole sûr contre les adversaires actifs. Notons que dans les rounds suivants chaque message est signé par l'expéditeur (σ_i^j est la signature sur le message msg_i^j dans les tableaux A.6-A.8) et cette signature est vérifié (avec les nonces) par le récepteur avant d'exécuter le protocole.

Round 1: le chef de groupe, M_l fait une demande initiale signée ($INIT$) avec son identité, U_l et un nonce aléatoire N_l au groupe $\mathcal{M}$ (voir le tableau A.6 pour le contenu exact de message)

Round 2: Chaque M_i intéressé répond à la demande de $INIT$, avec son identité U_i, à un nonce N_i et à un secret blindeé g^{r_i} à M_l (voir le tableau A.6 pour le contenu exact de message)

Round 3: M_l rassemble tous les secrets aveugles reçus, élève chacun d'eux à son secret (r_l) et les annonce au groupe avec les contributions originales, c.-à-d. il envoie $\{u_i, N_i, g^{r_i}, g^{r_i r_l}\}$ pour tout $i \in \mathcal{M} \setminus \{l\}$.

Calcul de clé: chaque M_i vérifie si sa contribution et son nonce sont inclus correctement et obtient g^{r_l} par le calcul $(g^{r_i r_l})^{r_i^{-1}}$. La clef de groupe est alors

Round 1
$l \xleftarrow{r} \mathcal{M}, N_l \xleftarrow{r} \{0,1\}^k$
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^1 = \{INIT, U_l, N_l\}, \sigma_l^1\}$
Round 2
$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_l}\{msg_l^1, \sigma_l\} == 1)$, $r_i \xleftarrow{r} [1, q-1], N_i \xleftarrow{r} \{0,1\}^k$,
$U_i \longrightarrow U_l : \{msg_i = \{IREPLY, U_l, N_l, U_i, N_i, g^{r_i}\}, \sigma_i\}$
Round 3
$r_l \xleftarrow{r} [1, q-1]$,
$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)$ and N_l is as contributed
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^2 = \{IGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M}\setminus\{l\}}\}, \sigma_l^2\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} and N_i are as contributed
$Key = g^{r_l(1+\sum_{i \in \mathcal{M}\setminus\{l\}} r_i)}$

Table A.6: Initial Key Agreement authentifié

$$Key = g^{r_l} * \Pi_{i \in \mathcal{M}\setminus\{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M}\setminus\{l\}} r_i)}.$$

Les protocoles de IKA, JOIN et DELETE sont presentée dans les tableaux A.6, A.7 et A.8.

Analyse de Sécurité - Dans cette section, nous discutons les objectits de sécurité pour un protocole d'accord de clé de groupe, nous expliquons et définissons d'autres notions préliminaires, avant de fournir une preuve de sécurité de notre protocole contre un adversaire actif dans le modèle de sécurité de Katz-Yung [KY03] (basé sur le modèle de [BCP02]).

Proprietes de sécurité - Les propriétés suivantes de sécurité peuvent être identifiées pour une protocole de GKA.

1) **Securité de la clé**: la clef peut être calculée seulement par les participants du protocole.

2) **L'indépendence des clés** : la connaissance d'un ensemble des clefs de groupe

Round 1
$\forall i \in \mathcal{J}, r_i \overset{r}{\leftarrow} [1, q-1], N_i \overset{r}{\leftarrow} \{0,1\}^k,$
$U_i \overset{B}{\longrightarrow} \mathcal{M} : \{msg_i = \{JOIN, U_i, N_i, g^{r_i}\}, \sigma_i\}$
Round 2
$\forall i \in \mathcal{J}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)\ r_l \overset{r}{\leftarrow} [1, q-1], l' \overset{r}{\leftarrow} \mathcal{M} \cup \mathcal{J}$
$U_l \longrightarrow U_{l'} : \{msg_l = \{JREPLY, \{U_i, N_i, g^{r_i}\}_{\forall i \in \mathcal{M} \cup \mathcal{J}}\}, \sigma_l\}$
Round 3
$if(\mathcal{V}_{PK_i}\{msg_l, \sigma_l\} == 1), l \leftarrow l', r_l \overset{r}{\leftarrow} [1, q-1], \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{J}$
$U_l \overset{B}{\longrightarrow} \mathcal{M} : \{msg_l^2 = \{JGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l^2\}$
Calcul de la clé
$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} and N_i are as contributed
$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table A.7: Protocole Join/Merge authentifié

Round 1
$\forall i \in \mathcal{D}, U_i \longrightarrow U_l : \{msg_i = \{DEL, U_i, N_i\}, \sigma_i\}$
Round 2
$\forall i \in \mathcal{D}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1), r_l \overset{r}{\leftarrow} [1, q-1], \mathcal{M} \leftarrow \mathcal{M} \setminus \mathcal{D}$
$U_l \overset{B}{\longrightarrow} \mathcal{M} : \{msg_l = \{DGROUP, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l, \sigma_l\} == 1)$ and g^{r_i} and N_i are as contributed
$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table A.8: Protocole Delete/Partition authentifié

ne mène pas à la connaissance d'une autre clef de groupe en dehors cet ensemble (voir [BM03]).

3) **Forward Secrecy** : La connaissance d'un certain secret à long terme ne mène pas à la connaissance des clefs passées de groupe.

Le modèle de sécurité qu'on utilise pour prouver la sécurité de notre protocole est proposé dans [BCP02, KY03].

Nous notons que le modèle de sécurité de [BCP02, KY03] n'aborde pas la question des nœuds malveillants. Des participants de session peuvent être corrompus, mais seulement après que la session sur laquelle l'adversaire fera la requête **Test** est passé. En outre notre définition de la *forward secrecy* ne donne pas l'accès aux données internes d'un nœud (tous les secrets courts et toutes les données stockés par le participant) à l'adversaire. Seulement la clef à long terme est indiquée à l'issue de la requête `corrupt`. Cette définition est désigné parfois sous le nom de *weak forward secrecy* en littérature. On peut noter que la propriété de *strong forward secrecy* peut être trivialement réalisée (quoiqu'avec perte d'efficacité) dans notre protocole en appliquant le protocole de **IKA** chaque fois qu'il y a un changement de composition du groupe. Le protocole récemment proposé du **KLL** ([KSML04]) est le seul exemple connu (de notre part) d'un protocole dynamique d'accord de clef de groupe réalisant la propriété de *strong forward secrecy*.

Theorem A.1 *Soit P le protocole comme défini précédemment. Soit $\mathcal{A}$ un adversaire actif faisant q_{ex} requêtes **Execute**, et q_s requêtes **Send** aux participants, et prenant un temps t. Alors le protole P est un protocole de GKA sûr. Précisément:*

$$\mathrm{Adv}_P(t, q_{ex}, q_s) \leq \mathrm{Adv}_{DDH}(t') + |\mathcal{P}| * Succ_{\Sigma}(t') + \frac{q_s^2 + q_{ex} q_s}{2^k} + \frac{1}{|G|}$$

où $t' \leq t + |\mathcal{P}|(q_{ex} + q_s)(t_{exp} + t_{lkup})$, t_{exp} est le temps d'un exponentiation dans G, k est le paramètre de sécurité and t_{lkup} est le temps pour faire une recherche dans les tables L et Sessions, définies dans la preuve.

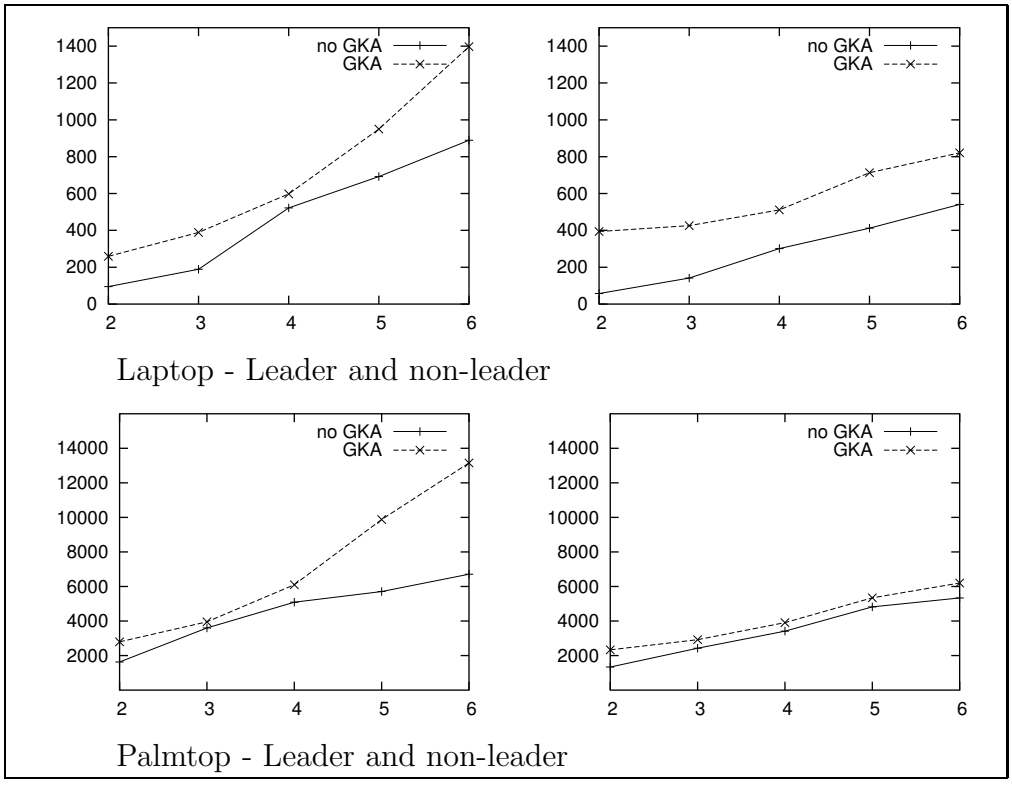


Figure A.3: Temps de calcul(en msec) par objets avec et sans GKA

Implementation - Dans cette section, nous présentons nos résultats d'exécution de la version non authentifiée du protocole dans le contexte d'un réseau ad hoc à simple saut, et nous étudions également comment le protocole authentifié peut être utilisé dans un réseau ad hoc à sauts multiples.

Pour évaluer le performance de notre protocole de GKA (sans authentification), nous l'avons incorporé dans le protocole de gestion de groupe de AdhocFS [BI03]. La gestion de groupe de [BI03] se compose de trois rounds de communication : $DISC$, $JOIN$ et $GROUP$. Létape de $DISC$ lance la formation de groupe par une requête aux participants intéressés. Chaque participant intéressé répond avec un message de $JOIN$. L'adhésion de groupe est définie et annoncée par le chef de groupe

(choisi aléatoirement) par le message de $GROUP$. La conception du nouveau protocole de GKA nous a permis d'accoller les données du protocole GKA aux messages de gestion de groupe, de même les contributions de membre vers le chef de groupe sont rassemblées pendant les messages de $JOIN$; tandis que le message de $GROUP$ diffuse le message du chef de groupe qui permet à chacun de calculer la clef de groupe. Ainsi aucun round additionnel n'est exigée pour dériver une clef de groupe, indépendamment de la taille de groupe. Il vaut la peine de mentionner que cela n'aurait pas été possible avec la plupart des protocoles présentés dans le tableau A.1, car les messages envoyés par des membres de groupe dépendent des messages envoyés par d'autres membres. Les temps de calcul sur un dispositif en l'absence et en la présence des procédures de GKA sont tracés dans la figure A.3. Les données montrées sont pour les ordinateurs portatifs se composants d'une installation expérimentale (Compaq 500 mégahertz sous Linux) et les palmtops (ipaq 400MHz de Compaq courant Linux familiar 07.). Toutes les contributions aléatoires pour la clef de groupe ont été choisies dans groupe de Diffie-Hellman d'ordre principal de 1024 bits. Le code a été écrit dans Java excepté la fonction d'élévation à une puissance qui utilise du code natif avec la bibliothèque GMP [Lib]. Les graphiques dans la figure temps de calcul 3.2 présente les temps (en millisecondes sur l'axe de Y) en fonction de la taille groupe avec et sans GKA. Il y a des courbes différents séparées suivant que le nœud est un leader/non-leader. Comem cést prévisible, le temps pour les membres non-chefs augmente (en utilisant le protocole de GKA) par un facteur presque constant (ordre de grandeur pour exécuter deux exponentiations dans un groupe de 1024 bits), alors que pour un chef il augmente linéairement à mesure que la taille de groupe augmente. Comme on s'attend à ce que la plupart des réseaux ad hoc se composent de dispositifs de puissance de calcul inégale, les appareils plus puissants (comme des ordinateurs portatifs) peuvent assumer le rôle du chef plus souvent. Ainsi notre protocole de GKA s'exécute bien dans un réseau ad hoc à simple saut, où le premier nœud publiant la demande de $DISC$ est choisi en tant que chef de groupe.

Nous présentons maintenant des détails d'exécution de notre protocole authentifié dans un réseau multi-sauts. Pour ce, nous considérons les réseaux ad hoc multi-sauts de taille moyenne (15 - 50 nœuds) avec un certain protocole de routage fournissant

la connectivité du réseau. Nous assumons également la présence d'un mécanisme de diffusion (broadcast) pour inonder le réseau. Pour de plus grands réseaux, la dérivation d'une clef de groupe de la contribution de chaque nœud peut prendre un temps considérable, à moins que seulement certains nœuds soient choisis pour participer au protocole.

Élection d'un chef de groupe : le protocole de GKA que nous proposons est asymétrique en ce qui concerne le rôle des membres dans le groupe. Un membre unique à chaque session initie le protocole et effectue un travail plus lourd que les autres. Bien que nous l'appelons le chef de groupe, il n'a aucune autorité spéciale ou de "capacités cryptographique" qui le rendent différent des autres. En fait l'avantage d'avoir un chef de groupe est qu'il facilite la définition de la composition d'un groupe avec un nombre minimum de messages. Par exemple, notre protocole peut être mis en application sans avoir besoin de prédéfinir un chef de groupe. N'importe qui peut émettre une message $INIT$ pour initier la formation d'un groupe, et les participants intéressés peuvent répondre à cette requête. De cette façon nous aurions un grand nombre de messages en diffusion dans le réseau, et nous finirions souvent avec plusieurs groupes. Mais si seulement un nœud dans le réseau diffuse la demande de $INIT$, on peut arriver à définir un seul groupe (hors du composant relié du réseau) avec beaucoup moins de trafic. Ce nœud particulier pourrait être changé (pour différentes raisons) aux sessions suivantes quand les clefs doivent être changées. Mais naturellement la question, est alors, comment choisir le nœud qui publie la première demande de $INIT$? Dans quelques applications, la réponse a pu venir de la nature de l'application elle-même. Par exemple, comme déjà observé, notre protocole de GKA s'intégré bien avec l'application d'AdhocFS [BI03] où il y avait déjà un besoin d'un chef de groupe pour vérifier la connectivité du réseau et l'adhésion du groupe. Mais pour les applications qui n'ont pas besoin d'un chef de groupe, la situation est un peu plus compliquée. L'élection du chef nécessite un protocole qui assure qu'exactement un chef soit élu à la fin du protocole (et que tous les autres nœuds connaissent le chef). Ces protocoles tendent à être fortement complexes. Beaucoup de contributions existes dans ce secteur et incluent [ORV94, Sin97]. Le meilleur protocole exige $n \log_2 n$ rounds pour décider d'un chef unique. Ainsi l'utilisation de tels protocoles sera trop chère en termes

de nombre de messages dans la plupart des scénarios. Les protocoles randomisés ou probabilistes d'élection de chef [GvRB00, MWV00] sont plus efficaces (bien que complexe) mais fournissent seulement certaines garanties probabilistes au sujet de l'unicité du chef. La dernière catégorie des protocoles semble davantage appropriée aux scénarios MANET. Nous présentons une approche beaucoup plus simple pour résoudre la question de l'élection de chef de groupe dans notre protocole. L'idée est de permettre à n'importe quel membre d'émettre en diffusion une demande de $INIT$, quand il détecte une absence du chef de groupe. Dans le cas de chefs multiples étant présents dans le réseau, le nombre de chefs est réduit lentement en utilisant des régles simples de résolution de conflit. Ainsi le protocole converge vers un chef simple mais probablement pas en une session simple mais pendant peu de sessions. Pendant cette période de convergence, le protocole continue à manipuler la dynamique de groupe. La résolution des conflits entre chefs de groupe et la gestion des changements de groupe peuvent être efficacement faites en exigeant du chef courant et des autres participants d'annoncer périodiquement leurs contributions au protocole. Ainsi nous considérons les messages suivants de notre protocole :

- $IGROUP$: ce message peut être émis par n'importe quel nœud (avec son identification et son nonce), quand il ne détecte pas la présence d'un chef de groupe. Il est également émis régulièrement par le chef de groupe courant, avec la composition courante du groupe, *les secrets aveugles* et *les réponses aveugles*. Quand un nouveau nœud entre en jeu, ce message sert de message de bienvenue par lequel il peut connaître la composition existante du groupe et le chef. Le nouveau nœud peut joindre le groupe en envoyant un message de $IREPLY$ avec sa propre contribution. À d'autres membres dans le groupe, le message de $IGROUP$ sert à contrôler leur vision de la composition du groupe et du chef de groupe. Ils peuvent également vérifier si leur contribution est bien incluse dans la clef de groupe. Pendant la période où il est chef, le chef de groupe est prié de réemettre (en broadcast) le message de $IGROUP$ avec une certaine période fixe T (détails plus tard). Ce message peut demeurer sans changement s'il n'y a aucun changement de groupe, ou peut aussi refléter les changements,

en ajoutant les contributions et les identifications de membres entrants, et en supprimant celles des membres sortants.

- $IREPLY$: ce message est utilisé par les membres qui joignent le groupe, pour réaffirmer leur présence dans le groupe. Ainsi les nouveaux membres peuvent répondre avec leur contribution tandis que les anciens membres peuvent simplement renvoyer ce message pour confirmer leur présence dans le groupe. Ils peuvent également changer leur contribution, avec ce message. Un membre de groupe est requis d'annoncer périodiquement ce message suivant une certaine périodicité.

- DEL: Ce type de message est envoyé par tout nœud désirant quitter le groupe. Comme nous ;'expliquons plus tard, le manque de messages consécutifs de $IREPLY$ d'un nœud donné est également considéré comme un congé par ce nœud. Nous maintenons cependant ce message pour avoir un mécanisme rapide pour gérer les membres quittant le groupe.

Ainsi avec ces messages, la formation de groupe et le calcul principal de groupe ont lieu comme suit : à l'arrivée dans un réseau, un nœud écoute s'il existe un message $IGROUP$ pendant une période fixe $k * T$ (où k et T sont connus a priori par tous les nœuds). Puisque des messages de $IGROUP$ peuvent être perdus, k doit être choisi de sorte que la probabilité que des transmissions successives de $k-1$ message de $IGROUP$ soient perdues soit petite. S'il ne reçoit pas un tel message vers la fin de cette période, il place un temporisateur aléatoire dans l'intervalle $[rtd, l * rtd]$, où rtd est le `round-trip delay` et l est un nombre entier. Si à l'échéance de ce temporisateur, aucun message de $IGROUP$ n'est reçu, le nœud émet son propre message de $IGROUP$, comprenant son identification et son nonce. L'idée de choisir cet intervalle est d'éviter à deux participants d'envoyer des messages de $IGROUP$ en même temps, après avoir détecté le manque d'un chef de groupe. Ainsi, par exemple, un nœud choisissant la période de temporisateur comme $7 * rtd$ est susceptible de recevoir le message de $IGROUP$ d'un nœud qui l'a choisi comme $5 * rtd$. Pour réduire encore la probabilité d'une collision, un nœud peut ajouter une petite décalage

(jitter) avant d'envoyer son message de $IGROUP$. Le paramètre l doit être fonction du nombre de nœuds dans le réseau et tel que la probabilité de deux nœuds choisissant le même nombre dans l'intervalle [1, l] est faible. D'autre part si un nœud reçoit le message d'un ou plusieurs $IGROUP$ avant l'échéance de ce temporisateur aléatoire, il doit avorter le processus d'envoi de message $IGROUP$. À la place, il doit choisir un message de $IGROUP$ auquel il peut répondre avec un message de $IREPLY$. Pour tous, une règle simple et même de décision (par exemple le chef de groupe avec l'identifiant minimal ou le groupe avec plus de membres) pourrait être employée pour décider à quel message de $IGROUP$ un nœud doit répondre. Ensuite après avoir choisi le groupe, le nœud envoie un message de $IREPLY$ au chef de groupe, avec ses contributions.

À la réception des messages $IREPLY$, le chef de groupe commence par mettre à jour le message de $IGROUP$ qu'il enverra à la prochaine échéance de la période T. Quand ce nouveau message de $IGROUP$ est émis, les membres déjà présents mettent à jour leur vision du groupe, et calculent la nouvelle clef de groupe (employant leur vieux secret). Les nouveaux membres peuvent voir que leur message de $IREPLY$ a été bien reçu et peuvent calculer la clef de groupe. Tous les membres de groupe doivent périodiquement annoncer leur message de $IREPLY$ avec la période T (et un petit délai, *jitter*). Ceci assure que le chef de groupe a une vision correcte et à jour des membres présents. Les contributions des membres et du chef de groupe doivent être changées souvent pour maintenir l'indépendance de clé et le secret de la clé. La propriété de *key independance* exige que la clef de groupe soit changée à chaque changement de la constitution du groupe, et pour la réaliser il suffit que le chef de groupe change correctement sa contribution à chaque modification de la constitution du groupe. Mais dans des réseaux fortement dynamiques et grands, ce peut être coûteux de calculer une nouvelle clef de groupe chaque fois que la consitution du groupe change. Des règles additionnelles pour changer la clef de groupe peuvent être employées, selon la dynamicité du réseau. Par exemple, la clef pourrait être changée si un certain nombre fixe de membres partent ou quand une certaine période fixe P expire.

Paramètre	Contrainte
durée pour changer de clé et/ou de chef, P	assez petit pour réfléter la dynamique du groupe
périodicité de réémission des messages $IGROUP$, T	assez petit pour réfléter la dynamique du groupe
Nombre d'intervalles d'attente d'un message $IGROUP$ avant d'émettre un $INIT$, k	assez grand pour s'assurer qu'un message ne s'est pas perdu
largeur de l'intervalle de *backoff* (mesurée *enrtd*), l	assez grand pour éviter des collisions lors de l'élection du chef
Round trip delay of the network, *rtd*	-

Table A.9: Paramètres du protocole

Les facteurs affectant le choix de ces paramètres sont détaillés dans le tableau A.9. L'échec du chef de groupe courant devrait automatiquement causer une auto-élection. La réélection du chef de groupe peut également être lancée à l'échéance de la période fixe P (la même que celle pour changer la clef). Les tableaux 3.15 et 3.16 donnent les procédures que doivent suivre le chef et un participant, respectivement.

Gestion des fusions et des scissions - L'émission périodique des messages de $IGROUP$ permet de gérer des fusions et des scissions dans le réseau. Ainsi, quand deux groupes ou plus avec leurs chefs respectifs fusionnent, la même règle simple que celle utilisée pour résoudre les conflits lors de l'élection de chef (taille de groupe ou identification inférieure du chef) est appliquée pour savoir quel chef continue. Ainsi les nœuds (y compris le chef de groupe) sont sûrs de recevoir les messages multiples de $IGROUP$, et décident quel nœud sera le nouveau chef. En cas de changement de chef, chaque nœud cesse d'envoyer des messages de $IREPLY$ (ou envie un message explicite de DEL) au vieux chef, et envoie en unicast sa nouvelle contribution de clef de groupe au nouveau chef. Si le chef de groupe ne reçoit pas un message de $IREPLY$ d'un nœud particulier pendant un nombre donné de périodes, le manque de réception de ces messages devra être géré comme la réception d'un message de DEL. En ce cas une mesure appropriée doit être prise par le chef de groupe qui recalcule un autre

```
Leader (k, T, l, rtd) {
    set-timer( t_1, T, timer-expiry-handle-fn1() );
    while (1) {
        listen(msg);
        if (msg.type == IGROUP)
            if (msg.senderID ≤ myID) exec(Participant(k, T, l, rtd, msg.senderID)) ;
            endif;
        else if (msg.type == IREPLY or DEL) updateIGROUP();
        endif;
    }
timer-expiry-handle-fn1() {
    reset-timer(t_1);
    send-msg(IGROUP);
}
}
```

Table A.10: Pseudo-code pour le chef de groupe

message de $IGROUP$.

Sécurisation du routage dans réseaux Ad hoc (Chapitre 4)

Les fonctions de routage dans les réseaux ad hoc mobiles sont effectuées par tous les nœuds disponibles dans le réseau. C'est tout à fait différent de la situation dans les réseaux traditionnels où seulement des nœuds dédiés participent au processus de routage. Cette participation par le réseau entier au processus de routage le rend plus efficace, mais en contrepartie elle le rend plus difficile à securiser. C'est non seulement vrai parce que maintenant le nombre des nœuds disponibles pour monter des attaques est beaucoup plus élevé, mais également parce que, tous ne peuvent pas physiquement être à l'abri des adversaires. La présence d'un seul nœud malveillant dans le réseau suffit pour perturber le processus de routage. Comme déjà discuté, le but d'un adversaire peut être varié : a) forçer un certain itinéraire à passer par son

```
Participant(k, T, l, rtd, current-leader-ID) {
    if (current-leader-ID == infinity) {
        set-timer( t_1, k * T, timer-expiry-handle-fn1() );
        set-timer( t_3, T, timer-expiry-handle-fn3() );
        }
    else prepare-mesg(IREPLY, current-leader-ID);
    while (1) {
        listen(msg);
        if (msg.type == IGROUP) process-IGROUPmsg(msg);
        else if (msg.type == IREPLY) exec(Leader(k, T, l, rtd));
        endif;
    }

process-IGROUPmsg(msg) {
    reset-timer(t_1);
    if (msg.senderID ≤ current-eader-ID) {
        current-leader-ID = msg.senderID;
        prepare-mesg(IREPLY, current-leader-ID);
    endif;
}

timer-expiry-handle-fn1() {
    set-timer( t_2, l * rtd, timer-expiry-handle-fn2() );
}

timer-expiry-handle-fn2() {
    reset-timer( t_2);
    reset-timer( t_1);
    send-msg( IGROUP);
}

timer-expiry-handle-fn3() {
    if (current-leader-ID != infinity) send-msg(IREPLY);
}
}
```

Table A.11: Pseudo-code pour un participant non chef

intermédiaire, b) forçer un certain itinéraire à ne pas passer par l'intermédiaire d'un nœud de son choix ou c) simplement détériorer l'efficacité du routage. Et pour réaliser ces objectifs, l'adversaire peut monter des attaques *passives* et/ou des attaques *actives*. En outre l'adversaire peut être un nœud *intérieur* ou un nœud *extérieur*. Par un nœud *intérieur*, nous nous référons à un nœud qui est connu par les autres nœuds comme faisant partie du réseau et qui est éligible pour participer au processus de routage. Tandis qu'un nœud est dit *extérieur* quand il s'agit d'un nœud dont la présence est inconnue (et non acceptée) par d'autres nœuds dans le réseau. Le but d'un nœud interieur pourrait être d'obtenir quelques avantages sans être pris, alors que celui d'un nœud d'éxterieur pourrait être simplement de rester caché et de continuer à monter des attaques. La mobilité des nœuds et la nature moins fiable du réseau rend la détection des nœuds *exterieur* très difficile. Dans les sections suivantes, nous discutons le travail antérieur effectué dans la sécurisation de routage des réseaux ad hoc et puis nous présentons un nouvelle primitive cryptographique, une signature à verificateur designé aggreggée, et nous exposonss comment l'employer pour sécuriser le routage dans les réseaux Ad Hoc.

Les travaux précédents : dans la littérature, on trouve des mécanismes cryptographiques et non-cryptographiques pour réaliser le routage sûr dans les réseaux ad hoc. Les mécanismes non-cryptographiques incluent l'utilisation d'information de localisation d'un nœud [CY02], et l'utilisation des plusieurs itinéraires redondants, entre deux nœuds, pour réaliser la robustesse [PH03a]. Nous discutons dans cette thèse seulement les travaux proposant des mécanismes cryptographiques pour sécuriser le routage. Les protocoles de routage pour les réseaux ad hoc peuvent être classés en trois catégories, à savoir, a) les protocoles proactifs de routage, b) les protocoles réactifs de routage et c) les protocoles hybrides de routage. Les exemples des protocoles proactifs sûrs de routage incluent le SEAD [HPJ02b] et SLSP [PH03b], alors que les exemples des protocoles de routage réactifs sûrs sont SRP [PH02], ARIADNE [HPJ02a], ARAN [SDL+03], SAODV [Zap] et endairA [BV04]. Le protocole ZRP [zrp] est un exemple de protocole hybride sûr de routage. Tandis que les protocoles de routage proactifs fournissent une meilleure performance dans les réseaux ad hoc avec une latence très basse, les protocoles réactifs de routage ont été montrés

plus performante pour les plupart des scénarios de réseau ad hoc [MM03, CM]. Ces derniers réagissent rapidement aux changements de topologie et maintiennent le coût de routage bas dans les périodes où il y a peu de changements dans le réseau. Par conséquent, nous détaillons les travaux existant dans le secteur des protocoles réactifs sûrs de routage. Nous rappelons le fonctionnement d'un protocole réactif de routage générique et comparons alors des protocoles réactifs sûrs connus pour les réseaux ad hoc.

Les protocoles réactifs de routage [PR99, dym] permettent le routage dynamique, multi-saut entre un ensemble de nœuds participants. Ils exigent que des messages soient échangés seulement quand un nouvel (ou non-existant) itinéraire entre une source et un nœud destinaire doit être découvert, ou quand un itinéraire existant échoue. En revanche, les protocoles proactifs de cheminement exigent l'échange périodique d'information de routage pour les itinéraires sans boucle et corrects. Ainsi en général trois genres de messages peuvent être définis pour des protocoles réactifs de routage:

- **Route Request Message (RREQ)** : c'est un message dont l'objectif est de découvrir un nouvel itinéraire entre une source et un nœud destinaire. Le nœud originaire de la requête émet un message de RREQ, indiquant sa propre identité et celle du nœud destinaire (avec des numéros d'ordre, etc.). Chaque nœud intermédiaire enregistre l'itinéraire jusqu'à la source et fait suivre au message ses voisins.

- **Route Reply Message (RREP)** : quand le message de RREQ atteint le nœud destinaire, (ou un nœud intermédiaire connaissant un itinéraire vers nœud destinaire), il répond avec un message de réponse d'itinéraire (RREP) vers le nœud source. Chaque nœud intermédiaire recevant ce message enregistre l'itinéraire au nœud destinaire, et fait suivre le message au prochain nœud sur l'itinéraire (établi par le message de RREQ) vers la source. Quand le nœud de source reçoit le message de RREP, des itinéraires ont été établis dans les deux directions entre la source et le nœud destinaire.

- **Route Error Message (RERR)**: Ce message est produit par un nœud quand il reçoit un paquet pour un itinéraire qui n'est plus disponible. Il oriente ce message vers la source du paquet. Ainsi le nœud de source peut relancer la découverte d'itinéraire en produisant un nouveau message de RREQ.

Les exemples de ce genre de protocoles incluent AODV [PR99], DSR [dsr] et DYMO [dym].

SRP: Le protocole SRP (Secure Routing Protocol) [PH02] est conçu comme une extension de protocoles réactifs de routage, et compte sur la disponibilité d'une association de sécurité (SA : security association) entre le nœud source (S) et le nœud destinaire (T). Un exemple de SA est une clef symétrique secrète K_{ST} dérivée en utilisant les clefs publiques de S et T. Ainsi S et T peuvent authentifier les messages de l'un et de l'autre par l'intermédiaire de cette clef symétrique partagée, en utilisant un MAC. Les nœuds intermédiaires qui transmettent par relais la requête RREQ vers la destination ajoutent leur addresse IP au RREQ, avant de le réexpédier, mais n'ajoutent aucune information d'authentification. À la réception d'un RREQ, le nœud destinaire vérifie l'intégrité et l'authenticité du RREQ en calculant le MAC des champs de requêtre, et en les comparant à la valeur du MAC contenu dans l'en-tête SRP. Si le RREQ est valide, le destinataire lance une réponse RREP où elle met le MAC du chemin accumulé. Il est nécessaire que les RREPs prennent l'itinéraire renversé pour obtenir de nouveau à la source.

Le protocole SRP souffre de l'attaque d'empoisonnement de cache d'itinéraire, *cache poisoning* : n'importe quel nœud intermédiaire qui expédie le RREQ pourrait ajouter beaucoup de nœuds faux au message. Ceci rend la probabilité de cet itinéraire d'être choisi comme itinéraire vers T par S très basse. C'est parce qu'il n'y a aucune authentification des messages point à point. SRP n'est pas non plus immunisé contre l'attaque du ver de terre (Wormhole).

ARIADNE: le protocole ARIADNE [HPJ02a] est un protocole réactif de routage, basé sur DSR, qui garantit que le nœud cible d'un procédé de découverte d'itinéraire peut authentifier l'initiateur, que l'initiateur peut authentifier chaque nœud intermédiaire

sur le chemin vers la destination, dans le message de RREP, et qu'aucun nœud intermédiaire ne peut enlever un nœud précédent dans la liste de nœud dans les messages de RREQ ou de RREP. L'hypothèse est que chaque nœud partage une clef symétrique avec d'autres nœuds, et que chaque nœud a une autre chaîne TESLA "key-chain" [PCST00, PCST01] dont une clef est connue à tous les autres.

Ainsi ARIADNE empêche des attaques de modifications de message mais à un coût certain. La taille des messages de routage augmente linéairement avec la longueur de l'itinéraire au fur et à mesure que tous les MAC sont apposés au message. En outre la synchronisation est une condition essentielle au fonctionnement d'AIADNE.

ARAN : le protocole ARAN [SDL+03] est un protocole de routage réactif, qui fournit l'authentification point à point des messages de routage, en utilisant des signatures numériques. Ainsi chaque nœud qui expédie le message RREQ ou le message RREP vérifie la signature du nœud précédent, l'enlève si elle est valide, et puis ajoute sa propre signature. Le protocole est sûr contre les modifications de message, mais il est prohibitivement cher à mettre en application dans les réseaux ad hoc.

SAODV: le protocole SAODV [Zap] emploie des signatures numériques pour authentifier le partie non-changeante d'un message de routage et des *chaînes de hachage* pour authentifier la partie mutable. En particulier le champ *hop count* d'un paquet est authentifié comme suit : le nœud source produisant le message de RREQ, choisit un nombre aléatoire *seed* et met $s = seed$ et $h^{MaxHopCount}(seed)$ dans le paquet de RREQ. N'importe quel nœud intermédiaire relayant le message de RREQ augmente le *hop count* par un et remplace la valeur précédente hash s par $h(s)$. Pour vérifier l'authenticité du *hop count* courant k, n'importe quel nœud peut vérifier si $h^{MaxHopCount}(seed)$ égale $h^{MaxHopCount-k}(s)$. Le protocole empêche des nœuds de diminuer le *hop count*, mais pas de le garder inchagé, ou de l'augmenter, ce qui aussi avoir comme conséquence des attaques.

endairA: Le protocole endairA [BV04] emploie la vérification de signature numérique sur chaque saut dans l'itinéraire, quand le message de RREP voyage vers la source. L'inconvénient est que la taille du message RREP dépend linéairement du nombre de

nœuds dans l'itinéraire. Le protocole suppose aussi que des attaques de trou de ver peuvent être détectées au niveau de la découverte de voisin.

Dans le tableau A.12, nous comparons les protocoles réactifs sécurisés. Nous mentionnons le mécanisme cryptographique utilisé par chaque protocole, la résistance aux attaques de manipulation (insertion ou suppression des nœuds dans le chemin), les coûts de calcul, et la taille des messages de chaque mécanisme de sécurité. Notons que le coût d'une opération asymétrique simple est habituellement beaucoup plus élevé que celui d'une opération symétrique simple. Le nombre de générations et de vérifications de signature et la taille de message sont donnés en fonctions de n, le nombre de nœuds dans l'itinéraire. SRP emploie seulement l'authentification d'extrémité à extrémité (seulement les nœuds source et cible vérifient la validité de la demande d'itinéraire), et est ainsi vulnérable aux attaques de manipulation par des nœuds intermédiaires. ARIADNE résiste à des attaques de manipulation, car chaque nœud vérifie la signature plus tôt et appose sa propre signature, mais exige une synchronisation lâche dans le réseau. ARAN résiste aux attaques de manipulation en exigeant de chaque nœud de vérifier la signature du nœud précédent, puis de mettre sa propre signature ayant enlevé la signature précédente. Ceci mène à de grandes tailles de message. SAODV exige des nœuds intermédiaires de vérifier la signature sur les champs non-mutables du message, et utilise des *hash-chains* pour résister à des attaques de manipulation. Mais il résiste à ces attaques seulement partiellement. Tandis que les nœuds sur un certain itinéraire ne peuvent pas diminuer le compte des sauts sur cet itinéraire, ils peuvent encore le garder identique ou l'augmenter. Le protocole endairA exige d'ajouter une signature numérique au message de RREP à chaque saut, ayant pour conséquence de grandes tailles de message. Plus loin dans cette thèse, nous montrons comment un protocole de routage réactif, utilisant notre proposition d'agrégation de signature à vérificateur désigné, permet l'authentification en utilisant des opérations cryptographiques symétriques, et en gardant constante de taille des messages de routage.

Tous les protocoles sont encore susceptibles d'être victimes des attaques de Wormhole. Les seuls protocoles connus comme résistant à cette attaque sont [PH03a]

	Mécanisme cryptographique	Résistance à la manipulation	Calucl		Taille des messages
			Gen	Ver	
SRP [PH02]	MAC	Non	$O(1)$	$O(1)$	$O(1)$
ARIADNE [HPJ02a]	$SA+TESLA$	Oui	$O(n)$	$O(n)$	$O(n)$
ARAN [SDL$^+$03]	DS	Oui	$O(n)$	$O(n)$	$O(1)$
SAODV [Zap]	DS + chaîne de hachage	Non	$O(1)$	$O(n)$	$O(1)$
endairA [Zap]	DS	Oui	$O(n)$	$O(n)$	$O(n)$
Ours [BHL06]	MAC	Oui	$O(n)$	$O(n)$	$O(1)$

SA: clé symétrique partagée entre toute paire de nœuds du réseau.
DS: Signature asymétrique.
Gen: Nombre de signatures (symétriques ou asymétriques) calculées sur la route
Ver: Nombre de signatures (symétriques ou asymétriques) vérifiées sur la route

Table A.12: Comparaison des protocoles de routage réactifs sécurisés

et [CY02]. Tandis que le [PH03a] se fonde sur la transmission des messages à travers des itinéraires multiples pour la détecter, [CY02] compte sur des systèmes de positionnement globaux précis.

Dans la prochaine section, nous présentons une nouvelle notion cryptographique qui peut aider à répondre à la question de l'authentification des messages de routage, et qui a l'avantage de reposer sur la cryptographie symétrique seulement. En outre, la taille des informations d'authentification ne dépend pas du nombre de nœuds sur l'itinéraire. Mais il reste faillible aux attaques de Wormhole. Cette primitive peut être employée avec n'importe quel protocole réactif de routage. Elle peut être également employée dans d'autres applications où des messages multiples doivent être authentifiés par un unique utilisateur.

Ici nous présentons une nouvelle primitive cryptographique pour authentifier efficacement des messages multiples envoyés par différentes entités à une entité unique. Nous montrons comment ce mécanisme cryptographique peut être employé pour authentifier des messages de routage dans des protocoles de routage pour les réseaux ad hoc (particulièrement protocoles réactifs de routage). Pour des protocoles proactifs

de routage, cette méthode peut seulement être employée si le protocole comporte l'authentification concourante des messages multiples des expéditeurs multiples par le même récepteur. Les notions de Aggregate Signatures [BGLS03] et Designated Verifier Signatures [JSI96] sont nécessaires pour comprendre la définition suivante :

Aggregate Designated Verifier Signatures : la notion d'Aggregate Designated Verifier Signatures (ou Ag_DVS) permet l'agrégation efficace des signatures pour le même vérificateur choisi. La motivation derrière une telle primitive est la suivante : bien que les signatures globales soit un concept très utile, les deux primitives existantes sont impraticables ou trop chères. Il semble qu'une structure additionnelle avec des permutations à trappe ou des couplages bilinéaires est un ingrédient essentiel pour pouvoir agréger des signatures. Mais nous prouvons ici que des signatures peuvent être très efficacement agrégées si elles sont toutes produites pour le même vérificateur désigné. Et il existe beaucoup d'applications qui peuvent utiliser ce dispositif (les protocoles réactifs de routage etant l'une d'entre elles, comme nous le montrons plus tard). La primitive d'Ag_DVS que nous proposons exige seulement de la cryptographie symétrique, et est ainsi tout à fait facile à implémenter dans les réseaux ad hoc. Ci-après, nous définissons formellement la primitive ag_DVS et nous en présentons une réalisation. Plus tard dans la section A, nous formalisons le modèle de sécurité, et nous analysons la sécurité de notre proposition.

Un schéma de signature agrégée à veficateur désigné (Ag_DVS) comporte les sept algorithmes suivants :

Ag_DVS.Setup : il prend comme entrée un paramètre k de sécurité et produit les paramètres *publics*. Ces paramètres publics sont des données implicites de tous les algorithmes suivants.

Ag_DVS.SKeyGen : il prend comme entrée un paramètre k de sécurité, et renvoie une paire (sk_A, pk_A) des clefs secrètes et publiques pour le signataire.

Ag_DVS.VKeyGen : il prend comme entrée un paramètre k de sécurité et renvoie une paire (sk_B, pk_B) de clefs secrètes et publiques pour le vérificateur.

Ag_DVS.Sign : il prend comme entrées un message m, (probablement) la clef publique

pk_B du vérificateur désigné B, et la clef secrète sk_A du signataire A. Le résultat est une signature σ_{AB}.

Ag_DVS.Verify : l'algorithme habituel de vérification prend comme entrées un message m, la clef publique pk_A du signataire, une signature σ_{AB} et la clef secrète sk_B du vérificateur désigné ; il renvoie 1 si la vérification est correcte, et 0 sinon.

Ag_DVS.Aggregate : cet algorithme prend comme entrée la clef publique pk_B du vérificateur et n tuples [3] $\{(pk_{A_i}, m_i, \sigma_{A_iB})\}_{1\leq i\leq n}$ de messages correctement signé par les différents utilisateurs A_i pour le même vérificateur indiqué B. Le résultat est une signature aggrégée Σ_B.

Ag_DVS.Ag_Verify : pour finir, l'algorithme pour vérifier l'exactitude d'une signature agrégée à vefificateur désigné vérificateur prend en entrée la liste de messages et des clefs publiques $\{(pk_{A_i}, m_i)\}_{1\leq i\leq n}$, la signature globale Σ_B, et la clef secrète sk_B du vérificateur B. Le résultat est 1 si la signature agrégée est correcte et est 0 sinon.

Construction de Ag_DVS de MAC : le cadre mathématique pour la construction est le suivant. Nous considérons un groupe $G = \langle g \rangle$ d'ordre premier q, engendré par un certain élément g; nous considérons la notation multiplicative. Nous supposerons que le problème calculatoire de Diffie-Hellman (CDH) est dur dans ce groupe.

Definition A.5 *Un algorithme $(t_{CDH}, \varepsilon_{CDH})$ résout le problème de CDH s'il reçoit comme entrée un tuple (g, g^a, g^b) avec $a, b \in \mathbb{Z}_q^*$ aléatoires, et produit la valeur g^{ab} avec la probabilité ε_{CDH}, en utilisant le temps d'exécution t_{CDH}.*

Nous supposons que notre groupe G est un groupe $(t_{CDH}, \varepsilon_{CDH})$-dur ; c'est-à-dire, il n'y a aucun algorithme dans lequel un algorithme qui $(t_{CDH}, \varepsilon_{CDH})$-résout le problème CDH dans G.

Notre proposition de construction d'Ag_DVS se sert d'un MAC. Par conséquent, nous supposerons également que nous avons un MAC (t', ε')-sûr, défini par les algorithmes MAC.Key, le MAC.Gen et MAC.Verify, avec l'espace de clés $\mathcal{K}$, et avec la longueur fixe

[3] On suppose, pour la simplicité de la notation et sans perte de généralité, que chaque utilisateur a signé seulement un message m_i.

ℓ pour les MACs produits par MAC.Gen (dans des propositions existantes des MACs, $\ell = 160$ est habituellement le cas).

L'arrangement d'ag_DVS que nous proposons marche comme suit :

Ag_DVS.Setup : si le paramètre de sécurité est k, alors un groupe $G = \langle g \rangle$ $(t_{CDH}, \varepsilon_{CDH})$-dur est choisi tel que l'ordre q de G est un nombre premier de taille 2^k. Une fonction de hachage cryptographique $H : G \to \mathcal{K}$ est également choisie.

Ag_DVS.SKeyGen/DVS.VKeyGen : pour chaque utilisateur U, un nombre aléatoire $sk_U \in \mathbb{Z}_q^*$ est choisi, et la clef publique associée est définie par $pk_U = g^{sk_U}$.

Ag_DVS.Sign : étant donné un message m, la clef publique pk_B d'un vérificateur désigné B, et la clef secrète sk_A d'un signataire A, la signature σ_{AB} est définie par

$$\sigma_{AB} = \mathsf{MAC.Gen}\big(m, H(pk_B^{sk_A})\big) \ \in \{0,1\}^\ell.$$

Ag_DVS.Verify : l'algorithme standard de vérification prend comme entrées un message m, la clef publique pk_A du signataire, une signature σ_{AB} et la clef secrète sk_B du vérificateur désigné. Il produit

$$\mathsf{MAC.Verify}\big(m, H(pk_A^{sk_B}), \sigma_{AB}\big).$$

Notons que le résultat de la vérification est correct, car $pk_A^{sk_B} = pk_B^{sk_A} = g^{sk_A sk_B}$.

Ag_DVS.Aggregate: l'entrée se compose des n tuples $\{ \ (pk_{A_i}, m_i, \sigma_{A_iB}) \ \}_{1 \le i \le n}$. La sortie est

$$\Sigma_B = \bigoplus_{1 \le i \le n} \sigma_{A_iB} \ \in \{0,1\}^\ell.$$

Ag_DVS.Ag_Verify: cet algorithme prend comme entrée la liste de messages et de clefs public $\{(pk_{A_i}, m_i)\}_{1 \le i \le n}$, la signature agrégée Σ_B et la clé secrète sk_B du vérificateur

B. La sortie est 1 si

$$\Sigma_B = \bigoplus_{1 \le i \le n} \mathsf{MAC.Gen}\big(m_i, H(pk_{A_i}^{sk_B})\big).$$

Notons que, en général, le vérificateur B doit recalculer toutes les signatures agrégées σ_{A_iB} pour vérifier l'exactitude d'une signature globale Σ_B (de longueur constante).

Les opérations les plus coûteuses dans la signature et la vérification se composent des élévations à une puissance dans le groupe, à savoir de calculer $pk_B^{sk_{A_i}}$ pour signer et $pk_{A_i}^{sk_B}$ pour la vérification. Ces opérations ne dépendent pas du message(s) signé, et ainsi peut être exécutées off-line, ramenant donc les opérations exigées en ligne à une génération ou à une vérification de MAC.

Analyse de sécurité: dans cette section, nous présentons le modèle de sécurité pour analyser la sécurité de notre proposition et ensuite nous présentons une preuve de sa sécurité.

Modèle de sécurité pour les schémas Ag_DVS

Nous considérons le niveau le plus élevé de la sécurité pour un arrangement d'Ag_DVS, qui est obtenu en combinant les modèles de sécurité pour des schémas standard de signature à vérificateur désigné, et des schémaux d'agrégation de signature. L'idée est qu'un attaquant $\mathcal{F}$ ne peut pas obtenir une signature globale valide pour un certain vérificateur désigné B, s'il ne connait pas la clef secrète sk_B de B, ou toutes les clefs sk_{A_i} secrètes des auteurs des signatures agrégées, ou les signatures agrégées elles-mêmes. On permet à l'attaquant de faire des signatures et des requêtes de vérification. Formellement, la sécurité est définie par le jeu suivant le $\mathtt{GAME}_2$ que l'attaquant $\mathcal{F}$ joue contre un challenger:

1. Le challenger prend un paramètre k de sécurité et exécute $\mathsf{Ag\_DVS.Setup}(k)$, $(sk_{A_1}, pk_{A_1}) \leftarrow \mathsf{Ag\_DVS.SKeyGen}(k)$ et $(sk_B, pk_B) \leftarrow \mathsf{Ag\_DVS.VKeyGen}(k)$.

2. L'attaquant $\mathcal{F}$ reçoit les clefs publiques pk_{A_1} et pk_B. Il peut exécuter les protocoles ag_DVS.SKeyGen et ag_DVS.VKeyGen tout seul pour obtenir d'autres

paires de clefs secrètes et publiques.

3. L'attaquant $\mathcal{F}$ peut faire trois sortes des requêtes :

   (i) une requête de signature pour message $\tilde{m}$; le challenger exécute $\tilde{\sigma}_{A_1B} \leftarrow \mathsf{Ag\_DVS.Sign}(\tilde{m}, pk_B, sk_{A_1}¿)$ et donne $\tilde{\sigma}_{A_1B}$ à $\mathcal{F}$.

   (ii) une requête de vérification pour les paires $(\tilde{m}, \tilde{\sigma}_{A_1B})$ de son choix ; le challenger renvoie à $\mathcal{F}$ la valeur $\mathsf{ag\_DVS.Verify}(\tilde{m}, pk_{A_1}, \tilde{\sigma}_{A_1B}, sk_B)$.

   (iii) une requête de vérification pour une signature globale $\tilde{\Sigma}_B$ correspondant à une liste des paires $\{(\tilde{pk}_{A_i}, \tilde{m}_i)\}$ de son choix ; le challenger renvoie à $\mathcal{F}$ la valeur $\mathsf{ag\_DVS.Ag\_Verify}(\{(\tilde{pk}_{A_i})de\tilde{m}_i\}, le\tilde{\Sigma}_B, sk_B)$.

4. À la fin, l'attaquant $\mathcal{F}$ produit une signature globale Σ_B pour une liste de paires $L = \{(pk_{A_i}, m_i)\}_{1 \le i \le n}$.

L'adversaire $\mathcal{F}$ réussit si:

(i) $\mathsf{ag\_DVS.Ag\_Verify}\big(pk_{A_1}, m_1, \ldots, pk_{A_n}, m_n, \Sigma_B, sk_B\big) = 1$; et

(ii) il y a au moins une paire $(pk_{A_1}, m_1) \in L$ tels qu'on n'a pas demandé m_1 par $\mathcal{F}$, dans une requête de signature dans l'étape 3(i) du jeu ci-dessus.

Si le temps de fonctionnement de $\mathcal{F}$ est t et sa probabilité de succès est ε, alors nous disons qu'il (t, ε)-casse la non-forgeabilité du schéma d'ag_DVS.

Definition A.6 *Un schéma d'Ag_DVS est* (t, ε)-secure *s'il n'y a aucun attaquant* $\mathcal{F}$ *qui* (t, ε)*-casse la non-forgeabilité du schéma.*

La propriété de la "source hiding" pour des signatures à vérificateur designé peut être appliquée à notre scénario d'Ag_DVS.

Definition A.7 (source hiding) *Un arrangement d'ag_DVS est* source hiding *s'il existe un algorithme* Simul *qui prend comme entrée seulement la clef secrète du*

vérificateur désigné et qui produit des chaînes binaires qui sont indistinguables (même avec la connaissance de toutes les clefs secrètes) des vraies signatures globales produites par de vrais signataires.

Preuve de sécurité : nous allons montrer que le schéma d'Ag_DVS construit dans la section précédente est sûr si le MAC utilisé est sûr, et si le problème CDH est dur dans le groupe G. La preuve est dans le modèle d'oracle aléatoire [BR93] pour la fonction de hachage $H : G \rightarrow \mathcal{K}$. Ceci signifie que nous supposons dans la preuve que cette fonction se comporte comme une fonction totalement aléatoire ; n'importe quel attaquant a accès à un oracle qui donne des réponses conformes, simulant le comportement idéal de la fonction H.

Theorem A.2 *Si le MAC utilisé est (t', ε')-secure et si G est un groupe $(t_{CDH}, \varepsilon_{CDH})$-sur, le schéma Ag_DVS proposé est (t, ε)-sûr, où*

$$\varepsilon = \varepsilon' + \varepsilon_{CDH} \quad and \quad t = \min\{t' - Q_H, t_{CDH} - Q_H\},$$

ici, Q_H est le nombre des requêtes que l'attaquant $\mathcal{F}$ contre l'Ag_DVS peut faire à l'oracle aléatoire H.

Preuve : supposons qu'il existe un certain attaquant $\mathcal{F}$ qui (t, ε)-casse la non forgeabilité de notre schéma d'Ag_DVS. Nous allons utiliser $\mathcal{F}$ comme une sous-routine pour construire un attaquant $\mathcal{A}$ qui casse la non forgeabilité du MAC utilisé. Nous expliquons comment cet attaquant $\mathcal{A}$ joue le jeu correspondant $\texttt{GAME}_1$ (rappel section 4.2.3).

Le challenger $\mathcal{A}$ prend un paramètre k de sécurité et exécute $K \leftarrow \mathsf{MAC.Key}(k)$. L'espace principal $\mathcal{K}$ est donné à $\mathcal{A}$.

À ce moment, le challenger $\mathcal{A}$ lance l'exécution de l'attaquant $\mathcal{F}$ supposé exister contre le schéma d'Ag_DVS. $\mathcal{A}$ va jouer le rôle d'un challenger contre $\mathcal{F}$ dans le jeu $\texttt{GAME}_2$ (rappel section A).

1. Tout d'abord, $\mathcal{A}$ prend un paramètre k' de sécurité et exécute $\mathsf{ag\_DVS.Setup}(k')$, $(sk_{A_1}, pk_{A_1}) \leftarrow \mathsf{ag\_DVS.SKeyGen}(k')$ et $(sk_B, pk_B) \leftarrow \mathsf{ag\_DVS.VKeyGen}(k')$.

2. $\mathcal{A}$ envoie à $\mathcal{F}$ les clefs publiques pk_{A_1} et pk_B.

3. L'attaquant $\mathcal{F}$ peut faire dans ce cas-ci quatre sortes des questions (parce que nous sommes dans le modèle de l'oracle aléatoire) :

   (i) une requête de signature pour le message $\tilde{m}$; pour répondre à cette question, l'attaquant $\mathcal{A}$ demande à son propre oracle (rappel définition de $\mathtt{GAME}_1$) le MAC du message $\tilde{m}$ en ce qui concerne la clef inconnue K. La valeur obtenue $\tilde{\sigma}_{A_1B} = \mathtt{MAC.Gen}(\tilde{m}, K)$ du MAC est transférée à $\mathcal{F}$; notons que l'attaquant $\mathcal{A}$ impose implicitement la relation $K = H\big(pk_B^{sk_{A_1}}\big) = H\big(g^{sk_B sk_{A_1}}\big)$.

   (ii) une requêtes de vérification pour les paires $(\tilde{m}, \tilde{\sigma}_{A_1B})$; dans ce cas-ci, l'attaquant $\mathcal{A}$ peut encore questionner son propre oracle de vérification de MAC, et renvoie à $\mathcal{F}$ le peu à partir dont il obtient $\mathsf{MAC.Verify}(\tilde{m}, K, \tilde{\sigma}_{A_1B})$.

   (iii) une requête de vérification pour une signature agrégée $\tilde{\Sigma}_B$ correspondant à une liste des paires $\{(\tilde{pk}_{A_i}, \tilde{m}_i)\}_{1\leq i\leq n}$ de son choix ; dans ce cas-ci, l'attaquant $\mathcal{A}$ peut employer la connaissance de la clef secrète sk_B pour recalculer
   $$\tilde{\Sigma}'_B = \bigoplus_{1\leq i\leq n} \mathsf{MAC.Gen}\big(\tilde{m}_i, H(\tilde{pk}_{A_i}^{sk_B})\big)$$
   et alors il renvoie à $\mathcal{F}$ la valeur 1 si $\tilde{\Sigma}'_B = \tilde{\Sigma}_B$, et 0 autrement.

   (iv) une requête de hachage $H(\tilde{y})$ pour un certain élément $\tilde{y} \in G$; pour répondre à cette question, l'attaquant $\mathcal{A}$ maintient une table des valeurs $\mathtt{TAB}$. Il procède comme suit : si $\tilde{y} = g^{sk_B sk_{A_1}}$, alors l'attaquant $\mathcal{A}$ s'arrête. Autrement, il recherche $\tilde{y}$ dedans $\mathtt{TAB}$; si la valeur est déjà là, alors $\mathcal{A}$ envoie à $\mathcal{F}$ la valeur $H(\tilde{y})$ stockée dedans $\mathtt{TAB}$. Si la valeur $\tilde{y}$ est nouvelle, alors $\mathcal{A}$ choisit au hasard la valeur $H(\tilde{y})$ dans $\mathcal{K}$, l'envoie à $\mathcal{F}$ et stocke la nouvelle relation dedans $\mathtt{TAB}$.

Supposons que $\mathcal{F}$ pose la question $H(g^{sk_B sk_{A_1}})$, avec une probabilité inférieure ou égale à ε_{cdh}. Alors nous avons que l'algorithme $\mathcal{F}$ peut résoudre le problème de CDH en temps $t + Q_H \leq t_{CDH}$, avec une probabilité plus grande que ε_{CDH}. Ceci nous donne une contradiction avec l'hypothèse que le groupe G est $(t_{CDH}, \varepsilon_{CDH})$-hard.

Nous pouvons supposer ainsi que $\mathcal{F}$ pose la question $H(g^{sk_B sk_{A_1}})$ avec une probabilité inférieure à ε_{CDH}. Ceci signifie que l'attaquant $\mathcal{A}$ ne s'arrête pas dans la phase des questions, avec une probabilité au moins $1 - \varepsilon_{CDH}$. Dans ce cas-ci, l'environnement de l'attaquant $\mathcal{F}$ a été parfaitement simulé par $\mathcal{A}$.

4. Par conséquent, par hypothèse, l'attaquant $\mathcal{F}$, dans le temps total t et avec la probabilité ε, donne une signature globale correcte Σ_b pour une liste des paires $L = \{(pk_{A_i}, m_i)\}_{1 \leq i \leq n}$ tels que il existe une paire (pk_{A_1}, m_1) telle que m_1 n'a pas été dans un requête de $\mathcal{F}$ ag_DVS.Sign à l'oracle, dans l'étape 3(i)

   Pour le reste de pzaires (pk_{a_i}, m_i), pour $i = 2, \ldots, n$, l'attaquant $\mathcal{A}$ calcule des valeurs $\sigma^{(i)}_{A_i B} = \mathsf{MAC.Gen}\big(m_i, H(pk_{A_i}^{sk_B})\big)$ en employant sa connaissance de la clef secrète sk_B de B, et il calcule la valeur $\sigma^{(1)}_{A_1 B} = \Sigma_B \oplus \bigoplus_{2 \leq i \leq n} \sigma^{(i)}_{A_i B}$.

   Par définition, c'est une signature a vérificateur designée valide pour le message m_1 et le signataire A_1, satisfaisant l'équation de vérification de MAC avec la cle privé $H\big(g^{sk_B sk_{A_1}}\big) = K$ (égalité implicite).

En résumant, l'attaquant $\mathcal{A}$ obtient en temps tout au plus $t + Q_H \leq t'$ un contrefaçon valide du MAC utilisé (note que $\mathcal{A}$ n'a jamais utilisé le message m_1 dans une requête à son MAC.Gen oracle). La probabilité de succès de $\mathcal{A}$ est la probabilité de l'événement suivant : $\mathcal{A}$ ne s'arrête pas et $\mathcal{F}$ produit un contrefaçon valide pour l'arrangement d'Ag_DVS. Cette probabilité est plus grande que $(1 - \varepsilon_{CDH})\varepsilon \geq \varepsilon - \varepsilon_{CDH} = \varepsilon'$.

Par conséquent, nous avons construit un attaquant $\mathcal{A}$ qui casse la non-forgeabilité du MAC en temps moins que t' et avec une probabilité de succès plus grande que ε'. Ceci contredit l'hypothèse que le MAC utilisé est (t', ε')-secure. Cette contradiction mène au résultat désiré : il ne peut pas exister un attaquant qui (t, ε)-casse la non forgeabilité de notre schéma d'Ag_DVS. En raison de ce théorème, nous avons que la

probabilité de casser notre schéma sera toujours $\varepsilon \leq \varepsilon' + \varepsilon_{CDH}$. Par conséquent, le schéma sera sûr (petite valeur de ε) si le MAC utilisé est sur (petite valeur de ε') et s'il est vraiment difficile de résoudre le problème de CDH dans le groupe G (petite valeur de ε_{CDH}).

Application au routage sécurisé : nous expliquons avec un exemple ci-dessous la façon dont le schéma ci-dessus peut être employé pour sécuriser des protocoles réactifs de routage. Considérons la figure A.4, où le nœud B produit d'un message de RREQ quand il veut trouver un itinéraire vers le nœud A_5. Le message est expédié comme représenté sur la figure. Quand le message atteint le nœud destinaire A_5, il répond avec un message de RREP. Notre schéma de schéma de signautre agrégée à vérfiicateur désigné nous permet d'inclure l'information d'authentification de tous les nœuds de la source à la destination. Chaque nœud qui prétend être sur l'itinéraire au nœud A_5 doit fournir une série appropriée de messages et une signature agrégée à vérificateur désigné sur ces messages pour convaincre le nœud B. Ainsi le nœud B est convaincu qu'il existe un itinéraire du nœud B au nœud A_5 par l'intermédiaire des nœuds A_1 et A_4 s'il peut obtenir des messages de chaque nœud disant leur position dans l'itinéraire et une signature agrégée pour confirmer l'authenticité de tous ces messages. Ainsi comme représenté sur la figure A.5, quand le nœud A_5 produit du message de RREP, il appose également sa signature $\sigma_{A_5 b}$ sur ce message, qui peut seulement être vérifié par nœud B. À la réception de ce message de RREP, le nœud A_4 ajoute son propres message, m_4 et fait le `xor` de sa signature $\sigma_{A_4 B}$ et de $\sigma_{A_5 B}$, et l'ajoute. La même opération est faite par le nœud A_1. À la réception du message RREP3, le nœud B vérifie les messages m_5, m_4, m_1 et une signature globale $\sigma_{A_1 B} \bigoplus \sigma_{A_4 B} \bigoplus \sigma_{A_5 B}$ sur ces messages. Ce que les messages contiennent réellement dépend du protocole particulier utilisé. Ce pourrait être le nombre de sauts (comme dans AODV) ou du l'information d'itinéraire (comme dans le routage de source).

Si un nœud malveillant (par exemple A_2) veut annoncer un itinéraire faux au nœud A_5, il devrait produire une signature globale $\sigma_{A_2 B} \bigoplus \sigma_{A_5 B}$, qui lui est impossible.

L'authentification des messages de RREQ a pu également être faite pour des protocoles de routage de source. Un inconvénient de la solution que nous proposons est que

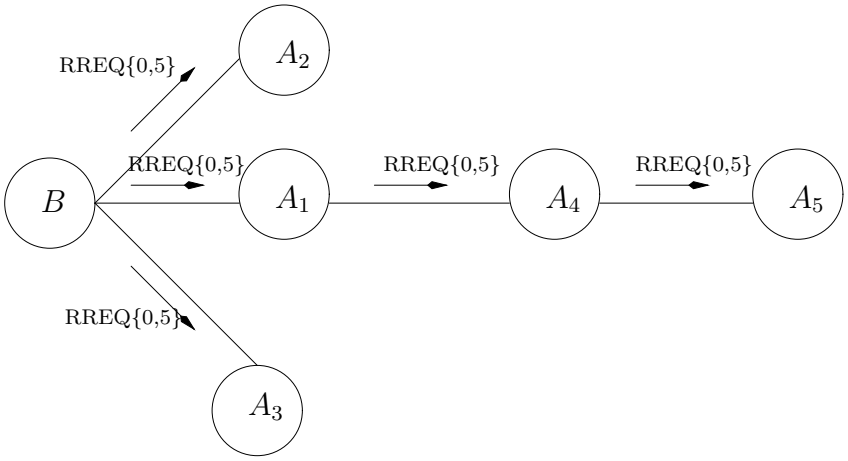
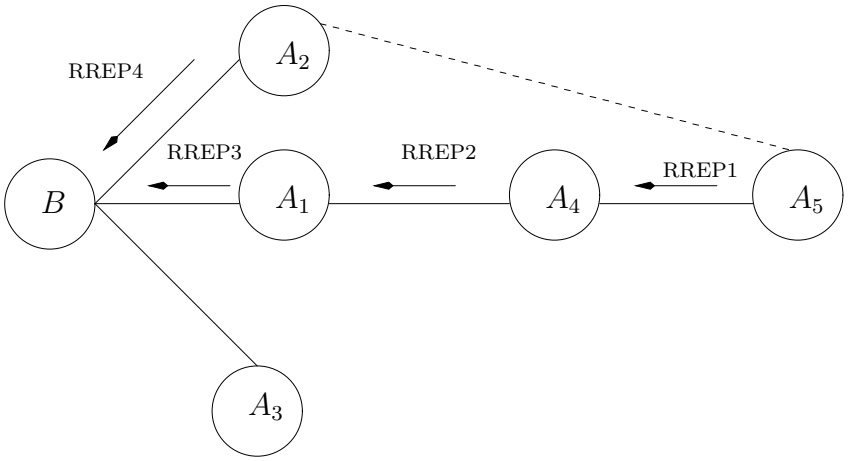


Figure A.4: Exemple de requête de route



RREP1 = { m_5 , σ_{A_5B} }
RREP2 = { m_5, m_4, $\sigma_{A_4B} \bigoplus \sigma_{A_5B}$ }
RREP3 = { m_5, m_4, m_1, $\sigma_{A_1B} \bigoplus \sigma_{A_4B} \bigoplus \sigma_{A_5B}$ }
RREP4 = { m_5, m_2, $\sigma_{A_2B} \bigoplus \sigma_{A_5B}$ }

Figure A.5: Exemple de réponse à une requête de route

les signatures à vérificateur désigné ne réalisent pas la propriété de non-repudiation. Intuitivement, étant donné une signature à vérificateur designé, le signataire et le vérificateur pourraient avoir produit cette signature aussi bien l'un que l'autre. Dans le contexte du cheminement réactif, cela implique qu'un nœud B qui a trouvé un itinéraire valide et authentifié jusqu'à un certain nœud A_j, ne peut pas convaincre n'importe qui d'autre de ce fait, parce qu'il pourrait avoir produit la signature par lui-même. Par conséquent, si un nœud différent C veut obtenir un itinéraire valide jusqu'à A_j, et s'il atteint le nœud B dans le processus, le nœud B ne peut pas convaincre C qu'il existe un itinéraire valide $B - A_j$ en employant sa signature globale. Par conséquent, les nœuds dans le chemin $B - A_j$ doivent signer un message encore un fois, cette fois ci avec le nœud C en tant que vérificateur designeé. Ceci a comme conséquence une perte d'efficacité si les nœuds sont tous honnêtes, mais au contraire c'est un point positif, car cela permet d'empêcher un nœud malveillant B' de tricher en prétendant qu'il existe un chemin valide $B' - A_j$ quand celui-ci n'existe pas en réalité.

Par conséquent, il n'est pas clair si la propriété publique de vérification réalisée par les signatures globales standard (avec la vérification publique) mènerait à une vraie amélioration d'efficacité dans la pratique. En fait, dans les réseaux ad-hoc mobiles où les nœuds changent leur position très souvent, la disponibilité des itinéraires doit être presque constamment mise à jour, et ainsi il est difficile d'imaginer qu'un "vieil" itinéraire $B - A_j$ puisse être accepter par un nœud C afin de construire un nouvel itinéraire $C - A_j$.

Conclusions et Perspectives (Chapitre 5)

La sécurisation des réseaux ad hoc est un challenge difficile. Le manque d'autorités globale de confiance, la puissance de calcul limitées des nœuds, la largeur de bande limitée et la nature dynamique du réseau rendent le problème de la sécurisation des réseaux ad hoc plus difficile que dans un réseau traditionnel. En même temps, il y a un besoin accru de coopération entre les nœuds pour pouvoir accéder à des services

sur le réseau. Les contraintes sur les dispositifs, dues à l'économie d'énergie, avec de plus une connectivité sporadique au réseau, entraîne que tous les services de sécurité doivent être distribués, de sorte que l'échec d'un nœud simple n'handicape pas tout le réseau. La recherche dans la sécurité des réseaux ad hoc s'est concentrée sur tous les services de sécurité de base comprenant le contrôle d'accès, l'authentification, la gestion des clés et la confidentialité. Un domaine de recherche très intense a été la sécurisation des protocoles de routage dans de tels réseaux. Tandis que dans Internet, la fonction du routage est exécutée par des nœuds choisis qui sont souvent (physiquement aussi bien que logistiquement) les nœuds les plus protégés du réseau, dans les réseaux ad hoc que cette fonctionnalité doit être exécutée par tous les nœuds. Ainsi le fonctionnement très basique du réseau peut par exemple dépendre d'un *palm-top* mobile avec de l'énergie limitée et aucune protection physique. D'où l'intêret accru du routage sécurisé dans les réseaux ad hoc. Dans cette thèse, nous étudions deux des secteurs de sécurité mentionnés ci-dessus, à savoir, la gestion principale des clés et le routage. Tous les deux sont des services de sécurité très basiques nécessaires pour fournir d'autres services de sécurité. Nous expliquons dans cette thèse pourquoi les protocoles principaux d'accord de groupe sont les plus appropriés parmi les principales techniques de gestion de clé dans les réseaux ad hoc. Les protocoles d'accord de groupe fournissent la capacité de dériver des clefs secrètes partagées, à partir des contributions de tous les nœuds dans les réseaux ad hoc dynamiques. Quelques protocoles existants de GKA exigent des opérations intensives en calcul pour chaque nœud, d'autres exigent de nombreux échanges de messages. Ceci rend ces protocoles peu adéquats pour des dispositifs contraints (avec la largeur de bande limitée) souvent présents dans un réseau ad hoc. Des protocoles efficaces existent, avec des procédures pour prendre en compte la dynamicité du réseau, mais ils exigent souvent un réseau bien structuré. Par un réseau bien structuré, nous voulons dire un réseau, qui a une structure logique des participants. Ceci est exigé pour être capable d'effectuer les communications synchrones exigées dans de tels protocoles. C'est une propriété difficile à réaliser dans les réseaux ad hoc en raison de la nature peu fiable des transmissions, ainsi que la composition variable dans le temps du réseau. De plus de tels protocoles, exigeant des communications synchrones, peuvent être stoppés quand il

y a défaillance d'un nœud. Nous proposons un nouveau protocole d'accord de clef de groupe, efficace dans le nombre de rounds de communication, et par le nombre de calculs. Il n'exige pas un réseau bien structuré, et l'échec d'un nœud n'empêche pas les autres nœuds de calculer la clef secrète partagée. Il exige qu'un nœud particulier dans le réseau exécute des fonctions différents des autres nœuds dans le réseau. Un tel nœud peut être choisi par un critère spécifique à l'application ou par un mécanisme d'auto-élection comme celui que nous proposons. De plus ce nœud spécialq peut être changé facilement et aussi souvent qu'exigé, ce qui permet de distribuer la charge dans tout le réseau. Le protocole surpasse la plupart des protocoles existants dans sa simplicité et dans son efficacité. Nous le prouvons dans le modèle standard, et nous avons tenté de mettre en application ce protocole dans un vrai scénario. Les domaines possibles des travaux futurs incluent l'étude de l'effet des nœuds compromis sur le secret des clefs secrètes partagées, et le perfectionnement du modèle de sécurité, plus concrètement l'étude de la possibilité d'avoir des nœuds compromis, et de la perte des messages dûe au milieu de communication. Les quelques protocoles proposés qui traitent du problème des nœuds compromis essaient 'empêcher contrôle de la clé par un de ces participants. La stratégie habituelle adoptée est d'ajouter un round de confirmation de la clé par lequel chaque participant du protocole de GKA est assuré de la valeur des clefs secrètes dérivées par les autres participants. Il reste à étudier l'effet des nœuds compromis sur des exécutions concourantes ou sur des exécutions inachevées du protocole. Le modèle de sécurité, doit aussi être renforcé pour tenir compte des nœuds internes malveillants. le modèle de sécurité doit aussi être améliorer pour prendre en compte la perte des paquets dans les réseaux ad hoc.

La deuxième partie de la thèse traite de la question du routage sécurisé dans les réseaux ad hoc. Les tentatives existantes de sécurisation du routage dans les réseaux ad hoc utilisent soit des mécanismes légers de sécurité (comme l'authentification d'extrémité à extrémité des messages de routage ou des chaînes de *hash* pour l'authentification nœud à nœud) dans un but d'efficacité, soit des mécanismes à sécurité forte (comme la vérification et l'authentification de signature de nœud à nœud) pour réaliser des niveaux plus élevés de sécurité. Tandis que les protocoles utilisant la première approche de sécurité sont vulnérables à une gamme entière des attaques,

les protocoles employant la deuxième approche ajoutent un fardeau lourd en calculs sur chaque nœud et augmentent de manière significative la taille des messages de routage. Nous présentons une nouvelle notion cryptographique : les signatures agrégées à vérificateur désigné, qui permettent que des signatures multiples, destinées à une même entité, soit efficacement agrégées. Un tel schéma permet de réaliser l'authentification forte de nœud à nœud des messages de routage, mais maintient en même temps une taille constante des messages de routage, et n'exige pas un effort calculatoire intensif pour chaque nœud. La contrainte est que la destination des messages de routage multiple doit être unique. Cela implique alors que cet arrangement peut être très efficacement appliqué à des protocoles réactifs de routage, mais qu'il peut ne pas être appliqu'e à des protocoles non-réactifs de routage. Dans ce dernier cas, les signatures de *agrégées* pourrait être un meilleur choix. Mais le seul schéma pratique de signature agrégée existant est gourmant en termes de calcul. Nous fournissons un modèle de sécurité pour étudier la sécurité des schéams d'ADVS. Nous proposons un schéma d'ADVS, qui est simple, comporte moins de calculs et à sécurité prouvée dans le modèle proposé. Un domaine possible de travaux futurs doit encore renforcer le modèle de sécurité afin de capturer efficacement les questions des participants malveillants, et des pertes de messages dans les réseaux ad hoc.

Bibliography

[802] IEEE 802.15 Wireless Personal Area Networks. http://www.ieee802.org/15/.

[ABIS05] D. Augot, R. Bhaskar, V. Issarny, and D. Sacchetti. An efficient group key agreement protocol for ad hoc networks. In *IEEE Workshop on Trust, Security and Privacy in Ubiquitous Computing*. IEEE CS Press, 2005.

[ABISar] D. Augot, R. Bhaskar, V. Issarny, and D. Sacchetti. A three round authenticated group key agreement protocol for ad hoc networks. *Elsevier Journal on Pervasive and Mobile Computing*, to appear.

[AG00] N. Asokan and P. Ginzboorg. Key agreement in ad-hoc networks. *Computer Communication Review*, 23(17):1627–1637, Nov 2000.

[BBM00] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in multi-user setting: Security proofs and improvements. In *Proceedings of Advances in Cryptology - EUROCRYPT*, volume 1807, pages 259–274. LNCS, 2000.

[BC04] E. Bresson and D. Catalano. Constant round authenticated group key agreement via distributed computation. In *Proceedings of Public Key Cryptography - PKC 2004*, pages 115–119. LNCS 2567, 2004.

[BCEP03] E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval. Mutual authentication and group key agreement for low-power mobile devices. In

Proceedings of the 5th IFIP-TC6 International Conference on Mobile and Wireless Communication Networks, pages 59–62. World Scientific Publishing, 2003.

[BCP02] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie Hellman key exchange under standard assumptions. In *Advances in Cryptology - EUROCRYPT '02*, pages 321–326. LNCS 2332, 2002.

[BD94] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Proceedings of Advances in Cryptology - EUROCRYPT*, volume 839, pages 275–286. LNCS, 1994.

[BEG02] J.S. Baras, L. Eschenauer, and V.D. Gligor. On trust establishment in mobile ad-hoc networks. In *Proceedings of 10th International Security Protocols Workshop*, pages 47–66. LNCS 2845, 2002.

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of Advances in Cryptology - EUROCRYPT*, volume 2656, pages 416–432. LNCS, 2003.

[BHL06] R. Bhaskar, J. Herranz, and F. Laguillaumie. Efficient authentication for reactive routing protocols. In *Proceedings of 2nd International Workshop on Security in Networks and Distributed Systems*. IEEE CS Press, 2006.

[BHLar] R. Bhaskar, J. Herranz, and F. Laguillaumie. Aggregate designated verifier signatures and application to secure routing. *Special Issue on Cryptography in Networks, International Journal of Security and Networks*, to appear.

[BI03] M. Boulkenafed and V. Issarny. AdHocFS: Sharing files in WLANs. In *2nd International Symposium on Network Computing and Applications*, pages 156–163. IEEE Computer Society, 2003.

[BKR00] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[Blo84] R. Blom. An optimal class of symmetric key generation schemes. In *Proceedings of Advances in Cryptology - EUROCRYPT*, pages 335–338. LNCS 209, 1984.

[BM03] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, 2003.

[BN03] C. Boyd and J.M.G. Nieto. Round-optimal contributory conference key agreement. In *Public Key Cryptography '03*, pages 161–174. LNCS 2567, 2003.

[Bon98] D. Boneh. The Decision Diffie-Hellman problem. In *ANTS-III: 3rd Algorithmic Number Theory Symposium*, pages 48–63. LNCS 1423, 1998.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93: Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[BT04] J.S. Baras and G. Theodorakopoulos. Trust evaluation in ad-hoc networks. In *Proceedings of ACM WISE*, 2004.

[BV04] L. Buttyan and Istvan Vajda. Towards provable security for ad hoc routing protocols. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 94–105. ACM Press, 2004.

[BW98] K. Becker and U. Wille. Communication complexity of group key distribution. In *Proceedings of 5th ACM Conference on Computer and Communications Security*, pages 1–6. ACM Press, 1998.

[BY93] M.J. Beller and Y. Yacobi. Fully-fledged two-way public key authentication and key agreement for low-cost terminals. *Electronics Letters*, 29:999–1001, 1993.

[CBH03] S. Capkun, L. Buttyan, and J.P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):1–13, 2003.

[Cha] D. Chaum. Private signatures and proof systems. United States patent 5493614.

[CM] S. Corson and J. Macker. Mobile ad hoc networking: Routing protocol performance issues and evaluation considerations. Internet RFC 2501.

[CPS03] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 197. IEEE Computer Society, 2003.

[CRVP01] K. Chandran, S. Raghunathan, S. Venkateshan, and R. Prakash. A feedback based scheme for improving TCP performance in ad hoc wireless networks. In *IEEE Personal Communications Magazine*, volume 8, pages 34–39, 2001.

[CY02] Stephen Carter and Alec Yasinsac. Secure Position Aided Ad hoc Routing. In *Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN '02)*, pages 329–334, November 4–6 2002.

[DB05] R. Dutta and R. Barua. Dynamic group key agreement in tree-based setting. In *ACISP*, pages 101–112, 2005.

[DDH+05] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security*, 8(2):228–258, 2005.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[dsr] The Dynamic Source Routing Protocol for Mobile Ad-hoc Networks. Internet Draft, draft-ietf-manet-dsr-10.txt.

[dym] Dynamic MANET On-demand routing. Internet Draft, draft-ietf-manet-dymo-03.txt.

[EG02] L. Eschenauer and V.D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of 9th ACM Conference on Computer and Communications Security*, pages 41– 47. ACM Press, 2002.

[ElG84] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of Advances in Cryptology - CRYPTO*, pages 10–18. LNCS 196, 1984.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[GvRB00] I. Gupta, R. v. Renesse, and K. P. Birman. A probabilistically correct leader election protocol for large groups. In *Proceedings of the 14th International Symposium on Distributed Computing (DISC)*, pages 89–103. LNCS 1914, 2000.

[HBC01] J-P. Hubaux, L. Buttyan, and S. Capkun. The Quest for Security in Mobile Ad Hoc Networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2001.

[HP04] Y-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. In *IEEE Security and Privacy magazine*, pages 28–39. IEEE, 2004.

[HPJ02a] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *MobiCom '02: Proceedings of the 8th annual international Conference on Mobile Computing and Networking*, pages 12–23. ACM Press, 2002.

[HPJ02b] Y-C Hu, A. Perrig, and D.B. Johnson. Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Fourth IEEE workshop in Mobile Computing Systems and Applications*, 2002.

[HV99] G. Holland and N. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proceedings of ACM Mobicom*, pages 219–230, 1999.

[i80] IEEE 802.11 Wireless Local Area Networks. http://grouper.ieee.org/groups/802/11/.

[ITW82] I. Ingemarsson, D. T. Tang, and C.K. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, 28(5):714–720, 1982.

[JSI96] M. Jakobson, K Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Proceedings of Advances in Cryptology - EUROCRYPT*, pages 142–154. LNCS 1070, 1996.

[JT87] J. Jubin and J.D. Tornow. The DARPA packet radio network protocols. In *Proceedings of the IEEE*, volume 75, pages 21–32. IEEE, 1987.

[KKFT02] S. Kopparty, S.V. Krishnamurthy, M. Faloutsos, and S.K. Tripathi. Split TCP for mobile ad hoc networks. In *Proceedings of IEEE GlobeCom*, pages 138–142, 2002.

[KPT04] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, July 2004.

[KS05] J. Katz and J.S. Shin. Modelling insider attacks on group key-exchange protocols. http://eprint.iacr.org/2005/163.pdf, 2005.

[KSML04] H-J. Kim, Lee S-M, and D.H. Lee. Constant-round authenticated group key exchange for dynamic groups. In *Proceedings of Advances in Cryptology - ASIACRYPT*, pages 245–259. LNCS 3329, 2004.

[KTC01] D. Kim, C.K. Toh, and Y. Choi. TCP-BuS: Improving TCP performance in wireless ad hoc networks. *Journal of Communications and Networks*, 3(2):1–12, 2001.

[KY03] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange - full version. In *Advances in Cryptology - CRYPTO '03*, pages 110–125. LNCS 2729, 2003.

[Lib] GNU Multi Precision Arithmetic Library. http://www.swox.com/gmp.

[LMRS04] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutaions. In *Proceedings of Advances in Cryptology - EUROCRYPT*, volume 3027, pages 74–90. LNCS, 2004.

[LS99] J. Liu and S. Singh. ACTP: Application controlled transport protocol for mobile ad hoc networks. In *Proceedings of IEEE WCMC*, pages 1318–1322, 1999.

[LS01] J. Liu and S. Singh. ATCP: TCP for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 19(7):1300–1315, 2001.

[LV05] F Laguillaumie and D. Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *Proceedings of Security in Communication Networks*, pages 107–121. LNCS 3352, 2005.

[LWB05] H. Lipmaa, G. Wang, and F. Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In *International Colloquium on Automata, Languages and Programming*, volume 3580, pages 459–471. LNCS, 2005.

[man] Mobile Ad-hoc Networks (manet). http://www.ietf.org/html.charters/manet-charter.html.

[MM03] Refik Molva and Pietro Michiardi. Security in ad hoc networks. In *PWC 2003, Personal Wireless Communications, September 23-25, 2003 - Venice, Italy*, 2003.

[MM04] B.S. Manoj and C. Siva Ram Murthy. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall, 2004.

[MNSS87] S.P. Miller, B.C. Neuman, J.I. Schiller, and J.H. Saltzer. Kerberos authentication and authorization system. Technical report, MIT, Cambridge, Massachusetts, 1987.

[MTI86] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distribution systems. *The Transactions of the IECE of Japan*, E69:99–106, 1986.

[MvOV96] A. J. Menezes, P. C. van Oorschot, and S. Vanstone. *HandBook of Applied Cryptography*. CRC Press, 1996.

[MWV00] N. Malpani, J. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks, 2000.

[MWW98] C.J. Mitchell, M. Ward, and P. Wilson. On key control in key agreement protocols. In *IEE Electronics Letters*, pages 980–981. IEE, 1998.

[NLKW04] J. Nam, J. Lee, S. Kim, and D. Won. DDH based group key agreement for mobile computing. http://eprint.iacr.org/2004/127, 2004.

[NS78] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21:993–999, 1978.

[ols] The Optimized Link state routing protocol version 2. Internet Draft, draft-ietf-manet-olsrv2-00.txt.

[OR87] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Sytems Review*, 21:8–10, 1987.

[ORV94] R. Ostrovsky, S. Rajagopalan, and U. Vazirani. Simple and efficient leader election in the full information model. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 234–242, New York, NY, USA, 1994. ACM Press.

[osi94] Open Systems Interconnection - Basic Reference Model. ISO Standard ISO/IEC 7498-1, 1994.

[PCST00] A. Perrig, R. Canetti, D. Song, and J.D. Tygar. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73. IEEE CS Press, 2000.

[PCST01] A. Perrig, R. Canetti, D. Song, and J.D. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium*, pages 35–46. The Internet Society, 2001.

[Per99] A. Perrig. Efficient collaborative key management protocols for secure autonomous group communication. In *Proceedings of International workshop on cryptographic techniques and electronic commerce*, pages 192–202. City University of Hong Kong Press, 1999.

[PH02] P. Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks. In *Communication Networks and Distributed Systems Modelling and Simulation Conference*. SCS Press, 2002.

[PH03a] P. Papadimitratos and Z. J. Haas. Secure message transmission in mobile ad hoc networks. *Ad Hoc Networks*, 1(3):193–209, 2003.

[PH03b] P. Papadimitratos and Z.J. Haas. Secure link state routing for mobile ad hoc networks. In *IEEE Workshop on Security and Assurance in Ad hoc Networks*. IEEE, 2003.

[PL00] J. Pieprzyk and C.-H. Li. Multiparty key agreement protocols. *IEE Proceedings - Computers and Digital Techniques*, 147(4):229–236, 2000.

[PR99] C.E. Perkins and E.M. Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100. IEEE CS Press, 1999.

[PW04] J. Pieprzyk and H. Wang. The key control in multi-party key agreement protocols. In *Proceedings of Workshop on Coding, Cryptography and Combinatorics '03*, pages 277–288. PCS, Birkhauser, 2004.

[SA99] L. Stajano and R. Anderson. The resurrecting duckling: Security issues in ad-hoc wireless networks. In *7th International Workshop on Security Protocols*. LNCS, 1999.

[SAHS03] K. Sundaresan, V. Anantharaman, H.Y. Hsieh, and R. Sivakumar. ATP: A reliable transport protocol for ad hoc networks. In *Proceedings of ACM Mobihoc*, pages 64–75, 2003.

[SBWP03] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. In *Proceedings of Advances in Cryptology - ASIACRYPT*, pages 523–542. LNCS 2894, 2003.

[SDL+03] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E. Belding-Royer. A secure routing protocol for ad hoc networks. In *IEEE International Conference on Network Protocols*, 2003.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[Sho] V. Shoup. Sequences of games: a tool for taming complexity in security proofs, manuscript.

[Sin97] G. Singh. Leader election in complete networks. *SIAM Journal on Computing*, 26(3):772–785, 1997.

[Sin00] S. Singh. *The Code Book: The Secret History of Codes and Code Breaking*. Knopf Publishing Group, 2000.

[SKM03] S. Saeednia, S. Kremer, and O. Markowitch. An efficient strong designated verifier signature scheme. In *Proceedings of 6th International Conference on Information Security and Cryptology*, pages 40–54. LNCS 2971, 2003.

[smf] Simplified Multicast Forwarding for MANET. Internet Draft, draft-ietf-manet-smf-01.txt.

[SSDW88] D.G. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A secure audio tele-conference system. In *Proceedings of Advances in Cryptology - CRYPTO*, pages 520–528. LNCS 403, 1988.

[Sta01] L. Stajano. The resurrecting duckling: What next ? In *8th International Workshop on Security Protocols*, pages 204–214. LNCS 2133, 2001.

[STW96] M. Steiner, G. Tsudik, and M. Waidner. Diffie-hellman key distribution extended to group communication. In *Proceedings of 3rd ACM Conference on Computer and Communications Security*, pages 31–37. ACM Press, 1996.

[TM04] Q. Tang and C. J. Mitchell. Efficient compilers for authenticated group key exchange. http://eprint.iacr.org/2005/366, 2004.

[TT00] W.-G. Tzeng and Z.-J. Tzeng. Round-efficient conference key agreement protocols with provable security. In *Proceedings of Advances in Cryptology - ASIACRYPT*, pages 614–627. LNCS 1976, 2000.

[vO93] P.C. van Oorschot. Extending cryptographic logics of belief to key agreement protocols. In *1st ACM Conference on Computer and Communications Security*, pages 232–243. ACM Press, 1993.

[YK02] S. Yi and R. Kravets. Key management for heterogeneous ad hoc wireless networks. In *Proceedings of 10th IEEE International conference on network protocols*, 2002.

[YK03] S. Yi and R. Kravets. MOCA: Mobile certificate authority for wireless ad hoc networks. In *Proceedings of 2nd Annual PKI Research Workshop*, 2003.

[Zap] M.G. Zapata. Secure ad hoc on-demand distance vector routing, ietf draft, draft-guerreo-manet-sadov-0 3.

[zrp] The Zone Routing Protocol for ad hoc networks. Internet Draft, draft-ietf-manet-zone-zrp-04.txt.