Efficient Key Agreement for Ad-hoc Networks

Master's Thesis

Maarit Hietalahti

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science
Espoo 2001

Teknillinen korkeakoulu
Tietotekniikan osasto
Tietojenkäsittelyteorian laboratorio

HELSINKI UNIVERSITY OF TECHNOLOGY	ABSTRACT OF MASTER'S THESIS

Author: Maarit Hietalahti

Name of thesis:
Efficient Key Agreement for Ad-hoc Networks

Date: May 23, 2001 **Pages:** 4 + 52

Department: Department of Computer Science and Engineering **Chair:** Tik-119

Supervisor: Professor Ilkka Niemelä
Instructor: Tuomas Aura D.Sc (Tech.)

Ad-hoc networks are communication networks that are constructed on the spot, often temporarily. Ad-hoc networks usually do not have a pre-existing infrastructure. The network consists of mobile terminals that connect by jumping from point to point via other peer devices. This causes the links to be unreliable and the network topology dynamic.

When bringing security to a system, a basic building block is the confidentiality of communication. This is achieved by encrypting the messages using a common symmetric key or asymmetric keys, of which the symmetric encryption is generally faster. Creating a common key for a group of communicators who all contribute to the key, is group key agreement.

The purpose of this thesis is to find a secure and efficient key agreement protocol for a group of communicating devices in an ad-hoc network. A special emphasis is on the suitability of the protocol to the difficult circumstances caused by the unreliable nature of ad-hoc network communications. This issue has received little attention in the literature. A thorough survey of the existing protocols from the ad-hoc point of view has been lacking and is now presented in this thesis. We found that none of the protocols in the literature is suitable for all ad-hoc network circumstances. Therefore, the main contribution of this thesis is a new key agreement protocol that can be implemented in the presence of unreliable connections and whose security and efficiency is comparable to the existing key agreement protocols. Our protocol, Arbitrary Topology Generalisation of Diffie-Hellman (AT-GDH), can be implemented in any network topology as long as at least one communication link exists to all parties. It uses the same security primitives as the protocols in the literature. It is optimal in the number of messages and nearly optimal in other efficiency measures.

Keywords: ad-hoc networks, key agreement
Language: English

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Maarit Hietalahti		
Työn nimi:	Tehokas avaimenluonti ad hoc -verkoissa		
Päivämäärä:	23. toukokuuta 2001	**Sivuja:**	4 + 52
Osasto:	Tietotekniikan osasto	**Professuuri:**	Tik-119
Työn valvoja:	Professori Ilkka Niemelä		
Työn ohjaaja:	TkT Tuomas Aura		

Ad hoc -verkko tarkoittaa tarpeen vaatiessa pystytettävää, usein tilapäistä tietoliikenneverkkoa, jolla ei ole keskitettyjä verkkopalveluja. Verkon muodostavat liikkuvat päätelaitteet, jotka ottavat yhteyksiä toistensa kautta. Tästä seuraa se, että yhteydet ovat epävakaita ja verkon rakenne muuttuu yhtenään.

Luottamuksellisuus, turvallisen kommunikaation tärkeä perusominaisuus, voidaan saavuttaa salaamalla viestit julkisella avaimella tai yhteisellä symmetrisellä avaimella. Näistä symmetrisen avaimen menetelmä on yleensä nopeampi. Ryhmälle viestijöitä on mahdollista saada yhteinen avain siten, että jokainen jäsen ottaa osaa avaimen luomiseen.

Tämän diplomityön tarkoituksena on löytää tehokas ja turvallinen yhteisen avaimen luontiprotokolla ad hoc -verkossa viestiville laitteille. Ad hoc -verkon tilapäisluonne aiheuttaa protokollalle erityisvaatimuksia, joita ei ole aiemmin kunnolla huomioitu kirjallisuudessa. Tässä työssä esitettävä yksityiskohtainen kirjallisuustutkielma olemassaolevista protokollista pyrkii korjaamaan tämän puutteen. Siinä osoitetaan, että mikään kirjallisuuden protokolla ei sellaisenaan sovellu kaikkiin ad hoc -verkon tilanteisiin. Siksi päätuloksena esitetään uusi avaimenluontiprotokolla, nimeltään AT-GDH, jota voi käyttää epävakaassa verkkotopologiassa. Protokollan turvallisuus ja tehokkuus on vertailukelpoinen kirjallisuuden protokollien kanssa. Se pohjautuu samoihin turvallisuusolettamuksiin kuin kirjallisuuden protokollat ja saavuttaa optimiarvon viestien määrän suhteen ollen lähes optimaalinen muiden käytettyjen mittareiden suhteen. Erityisesti tämän protokollan etuna on, että se toimii missä hyvänsä verkkotopologiassa, jossa kuhunkin osanottajaan on ainakin yksi linkki.

Avainsanat:	ad hoc -verkot, avaintenvaihto
Kieli:	englanti

Acknowledgements

This master's thesis was done in the Laboratory of Theoretical Computer Science. I wish to thank my supervisor, professor Ilkka Niemelä, and my instructor Tuomas Aura for their guidance. Without the huge amount of advice and encouragement of Tuomas this thesis could not have been written.

I thank my fiancé Tommi for his mental support. I also want to thank him for reading through my texts and fetching me home from work when I was exhausted from writing this thesis until the late hours.

I also want to thank my colleagues for their support, especially Silja Mäki.

Otaniemi 23.5.2001

Maarit Hietalahti

Contents

List of Figures

Chapter 1

Introduction

Ad-hoc networks are dynamic, peer-to-peer networks that do not have a pre-existing infrastructure. The communication security in ad-hoc networks is commonly based on a shared secret key known by the group of nodes that forms the network. The procedure for creating such a common secret for a group of communicating entities is called *group key establishment*. When the secret is created so that every node contributes to the secret, it is called *group key agreement*.

For example, when a group of people are in a closed meeting, they may want to form a private wireless network with their laptops in an ad-hoc manner. There may be no secure channel to connect the computers, and the participants may not have any common means to identify and authenticate each other digitally. This means that they originally do not share any secret keys nor have mutually verifiable public key certificate chains or access to a trusted key distribution center. An eavesdropper might listen to and interfere with their communication, even masquerade as one of them. They would need a group key establishment protocol, but the nature of ad-hoc networks sets certain special requirements for such a protocol.

Another example is a mobile military network, which is both security sensitive and easily exposed to security attacks. Not only the information passing in the network is confidential but the wireless traffic itself can reveal the location of a target to the enemy. The nodes, roaming in hostile environment with little physical protection, might be compromised, which increases the possibility of insider attacks. Finally, the network is highly dynamic, both in its topology and memberships. Nodes frequently join and leave the network, and a detection of compromised nodes necessarily also affects the trust relations. While most military networks still have hierarchical structures, it is increasingly necessary to form new networks ad-hoc on the battlefield.

Other examples of ad-hoc networks could include a wireless network of domestic appliances that serve the household and a network of wearable personal appliances such as mobile phones, laptops, mp3-players.

There are protocols in the literature for group key establishment, GDH.2 [STW96], Hypercube and Octopus [BW98], for example, but they have not been designed with the special nature of ad-hoc networks in mind. With this we mean that the protocols require the network topology to follow a pre-destined structure, a ring, for example. This is often impossible or impractical in the typical sparse connections of ad-hoc networks.

1.1 The Goal of This Thesis

The goal of this thesis is to find an efficient and secure protocol for group key establishment in the ad-hoc network environment. The ad-hoc network is assumed to be mobile and connections are assumed to be sparse. In other words, the links between nodes are few and the protocol cannot dictate the topology of the connection graph.

We will go through existing solutions and analyse their security, efficiency and suitability for ad-hoc networks. Furthermore, we will structure and identify problems that are specific to key exchange protocols in ad-hoc networks of arbitrary, dynamic topology. Keeping in mind the limitations of ad-hoc networks, we are especially interested in how flexibly the protocol can adapt to the arbitrary topology of an ad-hoc network. Additionally, we propose our own group key agreement protocol that is better suited to ad-hoc networks. This protocol can be applied in any network topology. We will also analyse its efficiency, security and reliability according to the criteria given in Chapter 3.

It is assumed that the neighbouring nodes have authenticated each other, or at least verified each other's group membership, so that all messages are known to come from authentic members of the group. The techniques for admitting new members to the group or removing them are not covered here. Instead, we refer to our group membership protocol in [MAH00]. This thesis will not treat the authentication or group membership protocol in detail. The security of the protocol is discussed mainly in the passive adversarial model as is the case with the original Diffie-Hellman key exchange.

1.2 The Organisation of This Thesis

The next chapter introduces several concepts needed in understanding the question of key establishment in an ad-hoc network environment, such as Diffie-Hellman key exchange and differences between key agreement and key transport. Chapter 3 explains the security properties and efficiency qualifications that we use as our criteria in comparing the protocols from the literature to our own. Chapter 4 gives an overview to the most interesting protocols from the literature. In Chapter 5 we present our own proposal AT-GDH for key agreement protocol in a network of arbitrary topology. This Chapter also evaluates and compares AT-GDH to the literature and presents some topics for future work. Chapter 6 presents the conclusions of this thesis.

Chapter 2

Background

This chapter introduces several concepts needed in understanding the question of key establishment in an ad-hoc network environment. Section 2.1 overviews the general situation in an ad-hoc network. Section 2.2 explains usual methods for building a shared secret, such as the Diffie-Hellman key exchange for two parties, and the differences of contributory and distributive key establishment. The need for authentication and some methods for it are mentioned in Section 2.3.

2.1 Ad-hoc Networking

Ad-hoc networks are dynamic, peer-to-peer networks that do not have a pre-existing infrastructure, i.e., they are constructed on the spot. Additionally, the parties involved might not have a common history. The ad-hoc networks are often temporary and the nodes can be mobile. The lack of infrastructure means a lack of central entities, fixed routers, name servers and such. Furthermore, the connections in ad-hoc networks are usually unreliable. The devices forming an ad-hoc network are often small and portable. Therefore, they do not have much memory or computational power and they are probably not tamper-resistant. Connections are formed by routing messages from point to point via other peer devices and not through dedicated router networks. An ad-hoc device can lose its connection to the rest of the network for several reasons: it can move out of the network's reach, it can run out of batteries or be compromised, or something similar can happen to the other devices that are connecting this device to rest of the network. Sometimes, the nodes have only sparse connections to each other. The network can change rapidly and occasionally even be partitioned. For these reasons, it is best to assume as little as possible about the network topology.

2.2 Key Establishment Protocols

The purpose of key establishment is to create a common key for a group of two or more participants to be used for encryption and authentication of their communications. For two participants, the *Diffie-Hellman key exchange* is often the most convenient choice. The multi-party case requires a generalisation of a two-way key exchange. This section contains some basic cryptographic terms and concepts that are not explained here. More information on them can be found in [MvOV97].

2.2.1 Diffie-Hellman Key Exchange

The Diffie-Hellman (DH) key exchange [DH76] is a way to create a symmetric key for two people who do not previously share a common secret. The parties (A and B) agree on a cyclic finite group G of order q, and a generator α for the group. After that, they choose secret exponents e_A and e_B. A computes its public key α^{e_A} in group G and sends it to B. Likewise, B sends α^{e_B} to A. Now, both parties know their own exponent and can raise the other party's public key to it, producing the shared key $\alpha^{e_A e_B}$. But a passive eavesdropper, who has not heard the exponents, cannot figure out the key. It should be noted that this protocol as itself does not pay attention to whom the message is from. Therefore further steps of authentication are necessary.

The protocols treated in this paper are generalisations of this protocol to the key exchange with more than two participants.

2.2.2 Key Distribution and Key Agreement

One of the biggest questions in choosing a group key establishment protocol is whether we choose a *distributory* or a *contributory* protocol (Figure 2.1). A contributory protocol means that all participants take equally part in the key generation and guarantee for their part that the resulting key is fresh. This is done by letting all parties $i (1 \leq i \leq n)$ choose a random element N_i, a nonce or a secret exponent as in the Diffie-Hellman key exchange. The key is then composed as a function of all key elements $K = f(N_1, \ldots, N_n)$. The requirements for the function f are the following.

- f is one-way for each argument N_i. This is because all N_i's should remain secret. If participant i is honest, then even a coalition of dishonest participants (a collusion) should not be able to calculate any information on N_i.
- everyone who has contributed a key share N_i can compute the key K from i and other information that they receive during the protocol
- it should not be possible to gain information of K without knowing at least one of the N_i's.

Contributory key establishment is also called *key agreement*.

Key distribution, on the other hand, means that the key is generated by one party and distributed to the other participants. This cannot be done without the help of a previously agreed-on secret that is used in encrypting the new session key, as is done for example in Shamir's no-key protocol. This secret can on the other hand be a very small one. Key distribution is also called key transport. There is also a method called key pre-distribution, whereby the key is completely determined by the previously agreed-on initial key material and no new secrets are created.

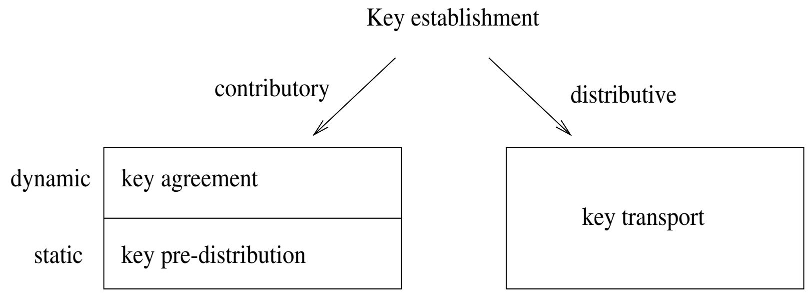


Figure 2.1: Key establishment methods

2.3 Message Authentication

As mentioned above, the basic Diffie-Hellman key exchange does not pay attention to ascertaining the origin of the protocol messages. Therefore it is possible to impersonate a genuine participant and, in the multi-participant case, to infiltrate the group of genuine participants. Consequently, it is necessary to authenticate the origin of the data to be a real member of the group. There are several ways of doing this and the method can be chosen independently of the group key establishment technique.

It should be noted that in this work, for the purposes of the key-establishment, only group membership verification is needed, not identity authentication. Furthermore, it is only necessary to verify the status of the neighbouring nodes, i.e., the ones with which the participant will have direct communication.

Authentication can be implemented, for example, by having for each node a signature key pair and a certificate that shows it is a member in the group (e.g. X.509 attribute certificate). Another way is to use the EKE-protocol [BM92] for authentication between nodes. In this context, the common secret must be sufficiently strong, bearing in mind the large number of links the ad-hoc network may have. As yet another possibility, we have presented [MAH00] a more elaborate group membership management scheme for ad-hoc networks that can be used in membership verification.

As there are many methods for message authentication that can be either separate protocols or interleaved with the key establishment, the efficiency of the contributory key exchange protocols in the literature is usually compared without authentication.

Chapter 3

Design Goals For Group Key Agreement Protocols

This Chapter explains what security properties the group key establishment protocol should have and what efficiency qualifications it should aim for. Section 3.1 deals with the special requirements the nature of ad-hoc networks poses to the key establishment protocols. Section 3.2 describes the security criteria for these kinds of protocols. Section 3.3 presents the lower bounds for the usage of data transfer capacity and time in a key agreement. These lower bounds have been proven in [BW98]. They apply to protocols where all parties contribute to the key and where a specific network topology may be required. This is often too strong a requirement for the typical sparse connections of ad-hoc networks but these figures can nevertheless serve as a guidelines for the lower bounds in ideal connectivity situations.

3.1 Creating a Common Key in Ad-hoc Network Environment

The limitations of ad-hoc network environment pose some drastic demands on the group key establishment protocols. First, a global broadcast is most probably out of the question, that is, it is not probable that an arbitrary node will have direct connections to all other participant nodes. But on some occasions, a local broadcast from a node to its neighbours is feasible. Also, no fixed topology, such as a ring or a star can be assumed. Consequently, most the protocols currently found in the literature either cannot be used at all or become inefficient.

The changing network topology demands the exchange to be as fast as possible since it is not certain that the contributing parties are connected to the network for a long time. It also means that protocols that put strong requirements on the network topology are difficult to implement.

The lack of infrastructure means that there are no third parties that can be trusted to calculate a random key safely and to distribute it. A lack of common history implies the lack of previously agreed shared secrets.

3.2 Security Properties

Here we list and describe the security properties that are desirable for a key establishment protocol in ad-hoc networks. Some of the properties, like secrecy and freshness of the key are essential while others, like key confirmation, may be needed in some applications but not in others. In any case, it is important to know which properties a particular protocol has and to compare them to the application requirements. More information on the security properties of a key establishment protocol and the cryptographic terms used here can be found, for example, in [MvOV97].

Operationality In the absence of active adversaries and communications errors, honest participant who comply with its specification always complete the protocol having computed a common key and the knowledge that only authentic members of the group share the key.

Unused and Fresh Session Key The session key should be fresh and unused at the end of the protocol. This means that the key should be new and that the key or elements of it should not be used for encryption or authentication during the protocol. This way, the adversary cannot have any previously encrypted or signed material for cryptanalysis and replay attacks.

Lack of Key Control The key should be generated from joint information instead of one participant having control over or being able to predict the value of the key. This is important when the parties do not trust each other completely.

Contributory Key Establishment Deciding the type of the protocol that best suites the problem area is the first step when building a group key establishment. We will now go deeper into the security implications of this decision in this particular case where we deal with the ad-hoc network environment.

Obviously, in a contributory key establishment we also have to assume that the network is not split at the moment of the group key agreement, i.e., that all nodes that contribute to the group key have a connection, direct or indirect, to each other. Key distribution can be performed over a longer time interval: delivering the key can wait until the absent nodes re-enter the network and are authenticated.

Contributory protocols are good when there are no previously agreed common secrets. The parties might not trust each other completely and need to be convinced that the generated key is fresh and random. In a protocol where every party generates a piece of the key, freshness of the resulting key is guaranteed when at least one participant generates a fresh random key element.

The lack of trusted third parties means that no one is trusted to calculate a random key safely and to distribute it. Even if such an entity can be appointed, it is a bottleneck in the protocol and it can be compromised or out of reach. A contributory protocol might not avoid these problems altogether, but it seldom has such single weak points that have network-wide consequences.

Secrecy An obvious requirement for a secure group key agreement is that it is secure against both active and passive eavesdroppers. A *passive eavesdropper* is an adversary that only listens to the messages without interfering with the protocol in any other way. The pieces of key material exchanged during the protocol should not reveal information that can be used to deduce bits of the group key. Neither should it be possible for an outsider to take advantage of the protocol in order to input extraneous key material that would compromise the key or to attain compromising information by pretending to be a genuine protocol participant. An adversary that uses these methods is called an *active eavesdropper*.

Security should be based on some generally accepted security primitives, like the Decision Diffie-Hellman assumption (DDH), which states that the distribution $\langle g^a, g^b, g^{ab}\rangle$ is polynomially indistinguishable from $\langle g^a, g^b, g^c\rangle$, where a, b and c are random numbers. This means that an eavesdropper should not be able to gain *any* bits of information on g^{ab} from g^a and g^b. For more information on the Diffie-Hellman assumptions see, for example, [Bon98].

The original Diffie-Hellman key exchange is secure against passive eavesdroppers only. Therefore we only consider the extensions of Diffie-Hellman in a passive adversarial model and assume that the case of an active adversary is dealt with in the authentication protocol. Security against active attackers is achieved by combining data origin authentication (that all messages come from authentic nodes of the network) with the freshness guarantees of the contributory Diffie-Hellman key exchange.

Implicit Key Authentication In the context of group key exchange, implicit key authentication means that a party can be sure that no-one outside the group can learn the key without the help of a dishonest participant.

Key Confirmation This means that after the key has been established, the participants are assured that all legitimate participants do share the same key. As this would require many all-to-all messages, which may not even be possible in a sparse connection ad-hoc network, achieving key confirmation is not practical.

Explicit Key Authentication Both implicit key authentication and key confirmation hold, i.e., all legitimate participants know the key and no outsiders do.

Resiliency This means that an active adversary cannot mislead honest participants as to the final outcome. For example, the adversary should not be able to convince an honest player that other participants know the secret key when, in fact, they do not. This is not as harmful as the actual compromise of the key.

Resistance to Known-key Attacks A compromise of past session keys does not allow a passive adversary to find out future session keys and does not allow impersonation by an active adversary in the future.

Independence of Long Term and Short Term Secrets This is important when there is an additional long term secret present, for example, private keys of a public-key algorithm or passwords used in authentication.

- *Perfect Forward Secrecy*: Compromise of a long term key does not compromise past session keys. This means that a session which has once been encrypted with a common session key remains secret unless that particular session key is compromised. Future sessions are nonetheless vulnerable if the same long term secret remains in use. Practical schemes for perfect forward secrecy are based on discarding temporary public keys.

- Protection of Long Term Secrets: Even if session keys are revealed to the attacker, it should not compromise the long term secrets such as master secret keys or the private keys of a public-key algorithm.

3.3 Lower Bounds for the Efficiency of a Contributory Key Establishment

In the case of a contributory key exchange, it is important to consider the communication and computational cost. K. Becker and U. Wille [BW98] studied systematically what can be achieved for the most common measures of complexity: the number of messages, exchanges and necessary rounds. The efficiency figures used here in comparing the complexity of the protocols also include the amount of local computation by a single node and total computation by all nodes together.

Becker and Wille present efficiency lower bounds separately for networks with and without global broadcast capabilities. These measures consider the complexity of key agreement protocols only. Any supporting mechanisms, like authentication or acknowledgement of messages, are not treated in these lower bounds. This is because we want the figures to be comparable to the ones used in the literature.

The measures used are explained here briefly. In all the tables, n is the number of nodes participating in the protocol. We assume a synchronous round model, i.e., that synchronous clocks tick at discrete intervals and every tick is one round.

Data Transfer Capacity Metrics:

Message This is a single packet of information (no matter how long) sent from one party to a single recipient.

DH-exchange This contains two messages exchanged concurrently by two single participants.

These measures keep account of the data transfer capacity that the protocol uses on one run. They are expressed as a function of the number of network nodes. Keeping this measure close to optimal is important especially in a situation where it is preferred to hide the existence of the ad-hoc network. Such a situation is probable in the case of a military network, for example, where a peak in the radio activity could alert the enemy to the presence of an ad-hoc network.

Time Complexity Metrics:

Synchronous round Every participant can send arbitrarily many packages concurrently within a single time tick (round) or receive arbitrarily many at the beginning of a round.

Simple round This means that every participant can send or receive at most one message per round.

These measures model theoretically the number of rounds, i.e., time, which one protocol run takes. Time is especially important in ad-hoc networks that change fast, usually due to node mobility.

As the processing time is in practice much larger than the time it takes to transfer the messages, the sending and receiving of messages are considered to be atomic events. When we use the synchronous rounds measure, several messages can be sent (practically) simultaneously and received simultaneously. This is usually a better model of the real situation than simple rounds, which is more accurate if the processing of a message takes less time than transferring it.

Other metrics:

(The number of) broadcasts Broadcast means a situation where one party sends the same message to all other participants simultaneously. We want to make a difference between global broadcast, where every node in the network actually has a direct connection to the broadcasting node, and *local broadcast* where the simultaneous message is sent only to the neighbours of the broadcasting node. As a global broadcast is not always possible in an ad-hoc network, it has to be substituted by routing the message via several steps of local broadcasts. This creates extraneous traffic in the network and causes loss of time. Therefore protocols using many global broadcasts are in practice less efficient than what the theoretical measures propose. Therefore we prefer to avoid protocols that require global broadcasts.

Local Computation In the Diffie-Hellman key exchange, the computation consists mostly of exponentiations. Therefore the amount of computation is evaluated by the maximum number of exponentiations a participant has to perform.

Total Computation This means the total number (the sum) of exponentiations performed by all participants. It should be noted that this does not necessarily equal n times the local computation as the average number of exponentiations per node may be lower than the maximum.

The Table 3.1 below contains the lower bounds for group key agreement protocols in networks without and with broadcast capabilities. These figures have been presented in [BW98].

Table 3.1: The lower bounds for the efficiency of key agreement

Lower bounds for efficiency without broadcast	
messages	$2n - 2$
DH exchanges	$2n - 4$ for $n > 4$
simple rounds	$\lceil \log_2 n \rceil$

Lower bounds for efficiency with broadcast	
messages	n
DH exchanges	n
simple rounds	$\lceil \log_2 n \rceil$
synchronous rounds	1

Chapter 4

Protocols in the Literature

In this chapter we study the existing solutions for group key establishment and evaluate their suitability for ad-hoc networks. The Section 4.1 goes through several group keying protocols that have been selected to represent the different approaches for this problem. All, except the tree-based key distribution by Burmester and Desmedt, are contributory protocols for the reasons described in Section 3.2.

4.1 Existing Group Key Exchange Protocols

Several solutions for extending the Diffie-Hellman key exchange to a multi-party key agreement have been proposed. One of the earliest (1982) is a rather unpractical protocol by Ingemarson et al. [ITW82]. This protocol assumes that the network nodes can be arranged as a ring. In the ring network, pieces of the key information are distributed along the ring so that the minimum amount of protocol rounds (without broadcast) is $n-1$, which is higher than optimal. This particular protocol is not secure either, since a passive eavesdropper can deduce the key if it is able to listen to all of the communications links simultaneously. In the rest of this section, we focus on protocols that are secure against all passive eavesdroppers and, if the messages are authenticated, also against active ones.

4.1.1 GDH.2 and Its Extensions

This protocol was presented by Steiner et al. [STW96]. It requires one broadcast message at the end of each protocol run. The network topology is a linear chain where the last node has broadcast capabilities (see Figure 4.1). In the last step of the protocol, the key pieces needed by each participant are broadcast to all parties. This could be changed by using n-1 constant size individual transmissions, but that would affect the efficiency measures.

GDH.2 protocol:

Initialisation: Let p be a prime and q a prime divisor of $p-1$. Let G be the unique cyclic subgroup of $\mathbb{Z}_p^*$ of order q, and let α be a generator of G.

Round i $(0 < i < n)$

1. M_i selects a random $r_i \in \mathbb{Z}_q^*$.
2. $M_i \to M_{i+1}$: $\{\alpha^{\frac{r_1 \ldots r_i}{r_j}} \mid j \in [1, i]\}, \alpha^{r_1 \ldots r_i}$

Round n

1. M_n selects a random $r_n \in \mathbb{Z}_q^*$.
2. $M_n \to ALL\ M_i$:
   $\{\alpha^{\frac{r_1 \ldots r_n}{r_i}} \mid i \in [1, n[\}$

The resulting key is $\alpha^{r_1 \ldots r_n}$.

The notation $M_i \to M_{i+1} : x$ means that the node M_i sends the message x to the node M_j.

This is an important protocol class since its security has been reduced to the security of the 2-way Diffie-Hellman. Several extensions have been proposed to this protocol ([STW98, AST98, AT99a, STW00, AT99b, AST00]).

4.1.2 The Hypercube Protocol

This protocol was presented by Becker and Wille [BW98] as an example of a protocol requiring the minimum number of rounds. Four nodes, which we shall call A,B,C and D, (as in Figure 4.2) are arranged in a square. They can create a shared key by just four Diffie-Hellman key exchanges: first A and B exchange keys in the usual way, and concurrently, C and D do the same. Next, A exchanges keys with C, using the 2-way key as the secret exponent. Concurrently, B and D do the same. (Figure 4.3). The result

$$K := \alpha^{\alpha^{ab}\alpha^{cd}}$$

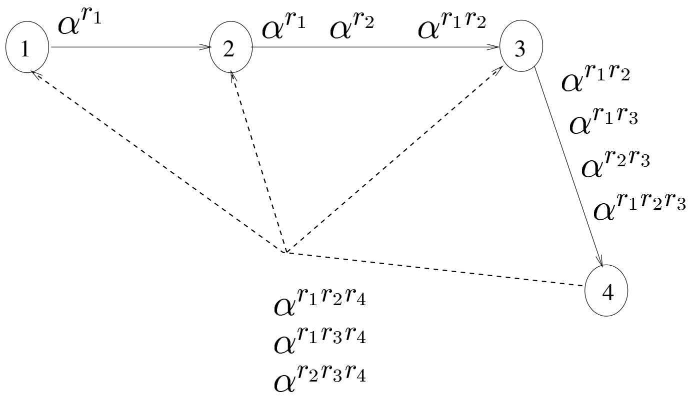


Figure 4.1: GDH.2 with n=4

is a key that is shared among all four participants. This is the hypercube protocol in the case where the dimension of the cube d equals 2.

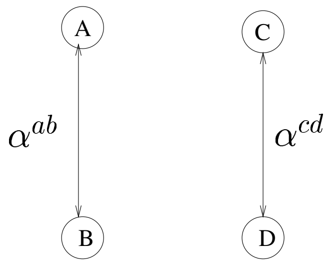


Figure 4.2: Round 1 of the protocol Hypercube.

The 4-way key exchange (square) can be generalised to a protocol for 2^d participants (d-dimensional hypercube) in d simple rounds. The next step is to continue the exchanges with another square of four participants using the created secret keys as the exponents. After d simple rounds of exchanges on the edges of a d-dimensional hypercube, all 2^d nodes have a common secret.

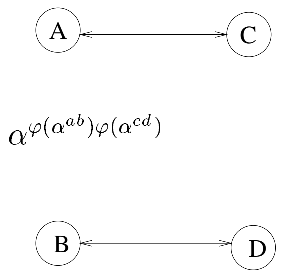


Figure 4.3: Round 2 of protocol Hypercube

Hypercube protocol for 2^d participants:

Initialisation: Let G be a finite cyclic group of order q, and let α be a generator of G. The participants are assumed to pick their secret exponents randomly from $\mathbb{Z}_q$. It is also assumed that there exists a bijection $\varphi : G \longrightarrow \mathbb{Z}_q$. Here the participants are identified with vectors of the d-dimensional vector space $GF(2)^d$ with basis $\vec{b}_1, \ldots, \vec{b}_d$.

Round 1

1. $\vec{v}$ selects $r_{\vec{v}}$
2. $\vec{v}$ performs a DH key exchange with $\vec{v} + \vec{b}_1$ using $r_{\vec{v}}$ as the secret exponent.

Round $i, (1 < i \leq d)$

1. $\vec{v}$ performs a DH key exchange with $\vec{v} + \vec{b}_i$ using the DH-key generated in round $i - 1$ as the secret exponent.

In [AG00], Asokan and Ginzboorg improved the 2^d-cube protocol by combining it with a password authenticated key exchange, and by adding ways to recover from node failures, to which this protocol is very sensitive.

4.1.3 The Octopus Protocol

This is another protocol by Becker and Wille [BW98]. It is an example of a protocol requiring the minimum number of messages. Here, the topology is such that four participants constitute the centre, a square (2^2-hypercube), and the rest of the participants form "tentacles", each of which is attached to any one of the central nodes as in Figure 4.4. First the tentacles do a Diffie-Hellman key exchange with their respective central nodes (Figure 4.5). After that, the central nodes perform the 4-party key exchange described in SubSection 4.1.2 using the products of the tentacle keys as their secret exponents (Figures 4.2 and 4.3). At this point, the central nodes have a key that is shared between all group members. Finally, the resulting key is distributed to the tentacles as in Figure 4.8.

Octopus protocol:

Initialisation: Let G be a finite cyclic group of order q, and let α be a generator of G. The participants are assumed to pick their secret exponents randomly from $\mathbb{Z}_q$. It is also assumed that there exists a bijection $\varphi : G \longrightarrow \mathbb{Z}_q$. Four participants, A, B, C and D, form the centre and the rest of the participants $P_1, \dots, P_{n-4}$, are each connected to one of them. I_X denotes the indexes of the nodes that are connected to the central node $X \in \{A, B, C, D\}$. I_A, I_B, I_C and I_D are pairwise disjoint.

Step 1

1. All $X \in \{A, B, C, D\}$ select a random r_X and all P_i when $i \in [1, n-4]$ select a random r_{P_i}.
2. For all $X \in \{A, B, C, D\}$ and all $i \in I_X$, X does a DH key exchange with P_i where r_X and r_{P_i} are the secret exponents of x and P_i, respectively. We denote the resulting keys $k_i = \alpha^{r_X r_{P_i}}$ when $i \in I_X$.

Step 2 A, B, C and D perform the 4-way key exchange described in SubSection 4.1.2 using the secret exponent values $a = K(I_A), b = K(I_B), c = K(I_C)$ and $d = K(I_D)$, where $K(J) := \Pi_{i \in J} \varphi(k_i)$ for $J \subseteq \{1, \dots, n-4\}$.

Step 3 A sends to all $P_i, i \in I_A$, the following two values:

$$\alpha^{K(I_B \cup I_A \setminus \{i\})} \text{ and } \alpha^{\varphi(\alpha^{K(I_C \cup I_D)})}$$

B, C and D send the respective two values to their tentacles.
The resulting key is

$$K = \alpha^{\varphi(\alpha^{K(I_A \cup I_B)}) \varphi(\alpha^{K(I_C \cup I_D)})}$$

Another version of Octopus was also mentioned in [BW98]: Octopus where the centre square (a 2-dimensional hypercube) was replaced by a hypercube of a greater dimension. This approach provides a trade-off between the low

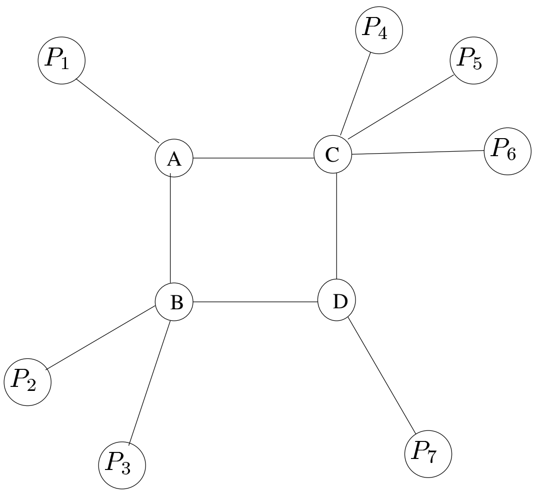


Figure 4.4: An example of the network topology in octopus protocol

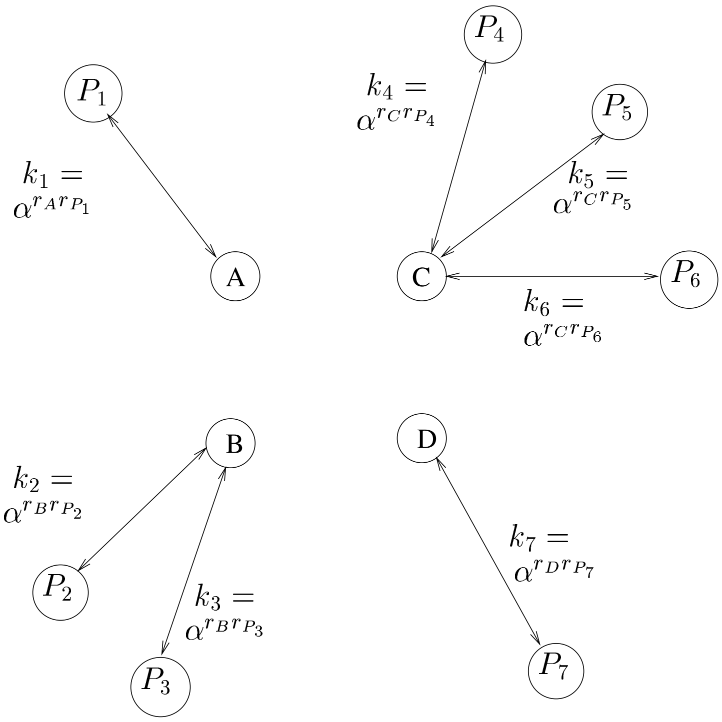


Figure 4.5: Octopus: round 1

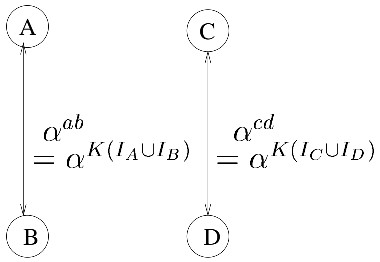


Figure 4.6: Octopus: round 2

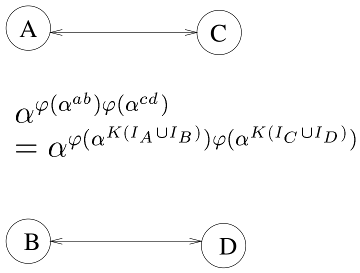


Figure 4.7: Octopus: round 3

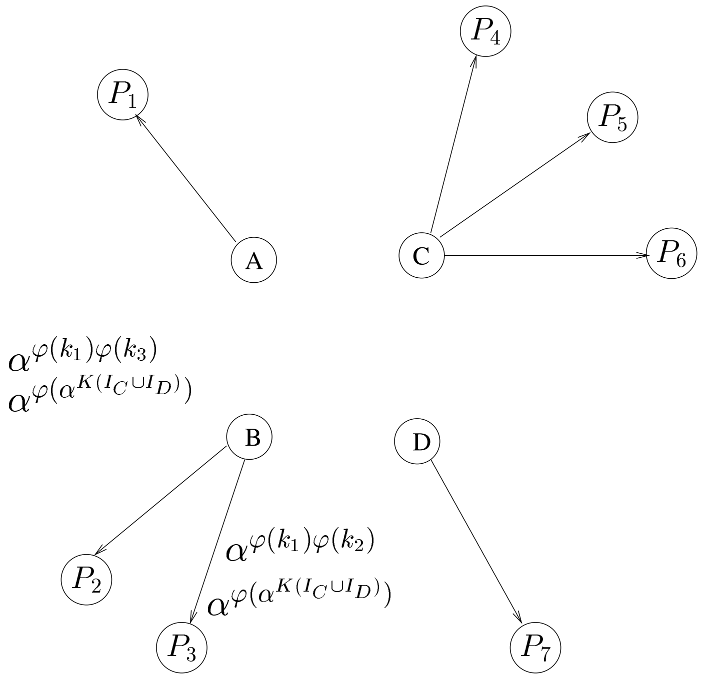


Figure 4.8: Octopus: round 4

number of rounds in Hypercube and low number of messages in Octopus at the cost of a more complicated network topology.

4.1.4 Binary Tree Based Protocol

Burmester et al. [BD94] present several conference key distribution systems based on public keys. They also extend these protocols to authentication and prove the security provided that Diffie-Hellman problem is intractable. From the ad-hoc network point of view, the most interesting protocol is the tree-based one. While this protocol is not contributory, we include it in the discussion because the tree structure provided ideas for our own protocol in Section 5.

Tree Based System by Burmester and Desmedt This is a key distribution protocol for a network whose topology is in the form of a binary tree. The root is a chair that generates the key and distributes it along the tree. The participants' contributions are only used in encrypting the session key while distributing it.

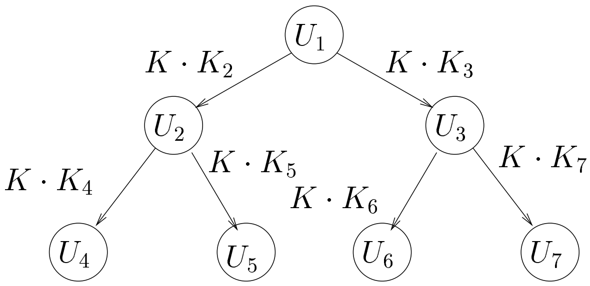


Figure 4.9: Tree based protocol: The final step

Tree based protocol:

Initialisation: Let p be a prime. $\alpha \in \mathbb{Z}_p^*$ generates a group $G \subseteq \mathbb{Z}_p^*$ of order q. The users $U_1, \ldots, U_n$ are labelled in such a way that the children of U_a are U_{2a} and U_{2a+1}. U_1 (the root) is the chair.

Step 1

1. For all $a \in \{1, \ldots, n\}$, U_a selects a random $r_a \in \mathbb{Z}_q^*$
2. U_a computes $z_a = \alpha^{r_a}$
3. $U_a \to U_{\lfloor a/2 \rfloor} : z_a$, if $a > 1$ and
   $U_a \to U_{2a} : z_a$, if $2a \leq n$ and
   $U_a \to U_{2a+1} : z_a$ if $2a + 1 \leq n$

Step 2 Each U_a computes

$K_a = z_{\lfloor a/2 \rfloor}^{r_a}$ if $a > 1$ and

$K_{2a+i} = z_{2a+i}^{r_a}$ for $i = 0, 1$, if $2a + i \leq n$

Step 3

1. U_1 selects a random session key $K \in G$
2. $U_1 \to U_{2+i} : Y_{2+i} = K \cdot K_{2+i}$, for $i = 0, 1$
3. Set $l = 0$

Step 4+l If U_a is at level l of the tree ($\lfloor \log_2 a \rfloor = l$) then

1. U_a decrypts Y_a in order to get K
2. $U_a \to U_{2a+i}$: $Y_{2a+i} = K \cdot K_{2a+i}$, $i = 0, 1$, if $2a + i \leq n$
3. Set $l = l + 1$

4.1.5 TGDH

In November 2000 (while this thesis was being written), Y. Kim, A. Perrig and G. Tsudik presented another protocol, TGDH ([KPT00]). Their group key agreement uses Diffie-Hellman key exchange in binary key trees. Actually, the paper does not define an *initial* key agreement protocol at all. They have not specified the steps taken and messages sent while forming the original group key. Instead, they just describe the structure of the resulting group key and concentrate on the dynamic properties, i.e. group management procedures such as join, leave, merge and partition.

However, the key structure in TGDH is very general, it can be used to describe the key structure of any contributive key agreement where the resulting keys are used recursively as the new secret keys. For example, the key structure of the protocol Hypercube is the same as that of TGDH with perfect binary tree where all leaves are at the bottom level.

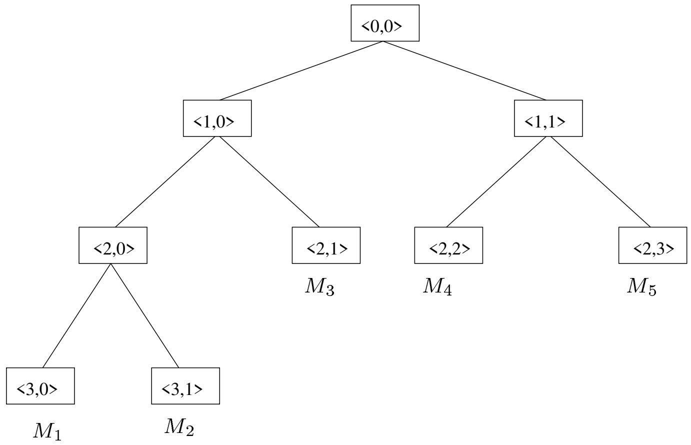


Figure 4.10: Notation for tree in TGDH

The Key Structure of TGDH

The nodes are denoted $\langle l, v\rangle$, which means the v-th node at level l in a tree, see Figure 4.10. (The addresses used here should not be confused with the universal addresses used in Chapter 5.1.) Each node $\langle l, v\rangle$ is associated with the key $K_{\langle l,v\rangle}$ and the blinded key $BK_{\langle l,v\rangle} = f(K_{\langle l,v\rangle})$ where the function $f()$ is modular exponentiation in prime order groups, analogous to the Diffie-Hellman protocol. α is the exponentiation base, p and q prime integers. M_i is the i-th group member.
The keys are computed recursively as follows:

$$\begin{aligned} K_{\langle l,v\rangle} &= (BK_{\langle l+1,2v+1\rangle})^{K_{\langle l+1,2v\rangle}} \bmod p \\ &= (BK_{\langle l+1,2v\rangle})^{K_{\langle l+1,2v+1\rangle}} \bmod p \\ &= \alpha^{K_{\langle l+1,2v\rangle}K_{\langle l+1,2v+1\rangle}} \bmod p \\ &= f(K_{\langle l+1,2v\rangle}K_{\langle l+1,2v+1\rangle}) \end{aligned}$$

The resulting group secret is $K_{\langle 0,0\rangle}$, the root's key.

It is clear that computing a key requires the knowledge of the key of one leaf and knowledge of the blinded keys of the siblings of each node on the path from the leaf to the root. As the paper does not specify the protocol steps where the blinded keys are sent, it is not possible to compare the efficiency of TGDH to other protocols. Therefore TGDH is not treated in the Section 4.2. It is possible to apply the protocol we present in Section 5 and achieve comparable results in efficiency.

4.2 Evaluation of the Protocols in the Literature

In this section, we compare efficiency properties of the protocols introduced in the previous section and analyse their impact on security. We evaluate also their suitability for ad-hoc network group key establishment. The efficiency numbers are given in tables for each protocol. None of the protocols achieve the optimum efficiency values by every measure. Some of them require a global broadcast capability, which is usually not available in ad-hoc networks. It may also be impossible to maintain a hypercube or even a ring topology or a binary tree of nodes in a changing network. The absent connections could be substituted by routing the messages via other connections, but this would slow the protocol down and generate extra messages. Consequently, the protocols from the literature either cannot be used at all or become inefficient in a network with arbitrary topology.

A protocol that can be realised in an arbitrary topology is presented in the next chapter.

GDH.2 and Its Extensions This protocol achieves the minimum in the number of messages and in Diffie-Hellman key exchanges. Unfortunately, this is achieved by requiring a broadcast capability for one of the nodes. The security properties are good since they have been reduced to the security of DDH. However, the broadcasting node has a central role and it is therefore a reliability risk. The list below presents the efficiency figures for GDH.2.

Efficiency parameters for GDH.2

messages: n participants form a ring which is proceeded through once, hence n messages.

exchanges: This protocol contains no actual exchanges, therefore the number of exchanges equals the number of messages, n.

simple rounds: The n messages are sent sequentially in n rounds.

syn. rounds: n, in analog to simple rounds.

local exponentiations: Participant i performs $i + 1$ exponentiations, the maximum is $n+1$ exponentiations for the last participant.

total exponentiations: $\sum_{i=1}^{n} i + 1 = \frac{(n+3)n}{2} - 1 = \mathcal{O}(n^2)$.

Hypercube This protocol has the minimal number of rounds. The number of messages is larger than the optimum. The number of DH exchanges is $\frac{nlog_2 n}{2}$, also above the minimum. But the main problem with this protocol is its vulnerability to node failures which is due to the strict requirements on the network topology. The number of local exponentiations is logarithmic to the number of nodes, and the total number of exponentiations is $O(n \log_2 n)$.

The list below presents the efficiency figures for Hypercube. Note that this protocol can be applied only if n is a power of two.

Efficiency parameters for Hypercube

messages: $n\lceil\log_2 n\rceil$. This equals two times the number of edges on a hypercube of dimension $d = \log_2 n$.

exchanges: One exchange is performed on each edge of the hypercube of dimension $d = \log_2 n$. So, the number of exchanges equals $\lceil\frac{n\log_2 n}{2}\rceil$

simple rounds: This equals d, the dimension of the hypercube. It can be seen so that each round adds one to the dimension of the cube. $d = \lceil\log_2 n\rceil$

syn. rounds: Same as simple rounds, $\lceil\log_2 n\rceil$. This is because the protocol does not contain the sending or receiving of multiple simultaneous messages from any node.

local exponentiations: Each node performs one exponentiation in the beginning and one on each round. This equals $1 + \lceil\log_2 n\rceil$

total exponentiations: All nodes perform the same number of exponentiations. The total number is n times local exponentiations, $n(1 + \log_2 n) = n + n\log_2 n = \mathcal{O}(n\log_2 n)$.

Octopus This protocol achieves the minimal number of 2-party DH exchanges as well, but this time without broadcast. The cost is in the number of messages, which is a bit higher than the minimum but nevertheless linear. As this protocol uses a hypercube in its centre, it inherits the vulnerabilities of Hypercube. Anyone of the four central nodes is a good target for an attack.

The efficiency parameters are shown in a list below. Note that the number of simple rounds and local exponentiations is calculated in optimal circumstances, i.e., when the tentacles can be evenly distributed between the four central nodes.

Efficiency parameters for Octopus

messages: First, $n-4$ tentacles perform a DH-exchange with their respective central nodes ($2*(n-4)=2n-8$ messages). The 2-dimensional hypercube in the center takes 8 messages. In the last step, all tentacles get one message ($n-4$). The total number of messages is $3n-4$.

exchanges: From above, we see that $n-4+4$ exchanges are done in the first 3 rounds. The final round contains $n-4$ messages. The total number of exchanges is $2n-4$.

simple rounds: The tentacles perform simultaneously the first exchange with their respective central nodes. Counting the exchanges separately and assuming that the tentacles are distributed evenly, we get $\lceil\frac{n-4}{4}\rceil$ simple rounds. In analog, the last messages have to be sent separately in $\lceil\frac{n-4}{4}\rceil$ simple rounds. The 2-dimensional hypercube takes 2 simple rounds. This gives a total of $2\lceil\frac{n-4}{4}\rceil+2$ simple rounds.

syn. rounds: This protocol takes 4 synchronous rounds: 1 in the beginning, 2 for the hypercube and 1 in the end.

local exponentiations: The maximum number of exponentiations per node is achieved in the central nodes, $2|I_X|+5$ where $|I_X|$ is the number of tentacles a central node has. If the distribution of the tentacles is optimal, this equals $2\lceil\frac{n-4}{4}\rceil+5$

total exponentiations: The total number $6n-4=\mathcal{O}(n)$ is computed by multiplying the number of tentacles ($n-4$) by their exponentiations (4) and adding this to the number of exponentiations done by the four center nodes ($2\sum_{X\in\{A;B;C;D\}}|I_X|+4*5=2n+12$).

Tree Based System by Burmester and Desmedt This protocol is not contributory. Hence, it cannot really be compared to the other protocols by the measures in Chapter 3. The root decides a key and distributes it along the tree, which makes the root an attractive attack target and a security risk. In a tree structure, the participants must also trust their parent nodes against denial-of-service attacks. If one node changes the key it has received from its parent, then all of its descendants will have a wrong key. The number of protocol rounds, $1+\lceil\log n\rceil$, is very good with this protocol.

Chapter 5

Key Agreement in Networks of Arbitrary Topology

In this chapter, we present our suggestion for key agreement in ad-hoc networks. Section 5.1 contains explanations of some definitions and terms used in the protocol description. Section 5.2 contains the construction of the spanning tree and the actual protocol description. In Section 5.3 we evaluate the security and efficiency of the protocol according to the criteria presented in Chapter 3. Issues that need to be taken into consideration when using the protocol in practice are discussed in Section 5.4. In Chapter 5.5 we present suggestions for future work.

5.1 Definitions

This section explains briefly the terms used here for describing graphs and trees. For a more thorough explanation on the theory of graphs see, for example, [Die97].

Definition 5.1.1 (Graph) *A* graph *is a pair $G = (V, E)$, where V is a finite non-empty set and $E \subseteq \{\langle x, y\rangle | x \in V, y \in V, x \neq y\}$. The elements of V are called* nodes *and the elements of E are called* edges.

Definition 5.1.2 (Tree) *An undirected graph $G = (V, E)$ is a* tree, *if it does not contain any cycles and is connected, i.e., a tree is a minimally connected graph. Minimal means that if a graph G is connected and, if e is any edge, then $G - e$ is not connected.*

A *rooted tree* is a tree where one node has been selected as the root. The *level number* of a node s is the length of the (unique) path from the root to s. The root has level number 0. The nodes on level 1 are the root's *children.* For a node s on level l, a node t on level $l+1$ is s's *child* if the path from s to t has length 1. Likewise, s is t's *parent. Siblings* are nodes that have the same parent. The *ancestors* of a node s are all nodes that lie on the path from s to the root (including the root). The *descendents* of s are all nodes that have s as their ancestor. A *leaf* is a node that has no children. All other nodes are called *branch nodes.*

The *height* of the tree h is the maximum distance from the root to a leaf. A rooted tree is *balanced* if every leaf is on level h or $h-1$.

A rooted tree is a k*-ary tree* if all of its branch nodes have k or less children. In a *perfect* k*-ary tree* every branch node has exactly k children. A binary tree is a k-ary tree with $k=2$.

Definition 5.1.3 (Spanning tree) *Let $G=(V,E)$ be a graph. A subgraph $G=(V',E')$ with $V'=V$ that is also a tree is a spanning tree.*

Proposition 5.1.4 *Every connected graph contains a spanning tree.*

Proof: A graph $G=(V,E)$ is connected but not a tree. Remove all cycles from the tree by removing an edge from each cycle until it is not possible to remove an edge without disconnecting one or more nodes from the graph. The resulting subgraph $G'=(V,E)$ is minimally connected, hence it is tree. It is also a spanning subgraph, because it contains all of G's nodes. Therefore, it is a spanning tree.

Calculations in a perfect k-ary tree For a perfect k-ary tree T of height h, the number of leaves l is less than or equal to k^h. If T is balanced, $h=\lceil log_k l\rceil$.

The number of nodes in a balanced perfect k-ary tree: $n \leq \sum_{i=0}^{h} k^i = \frac{k^{h+1}-1}{k-1}$
If all leaves are at the bottom level, $n=\frac{k^{h+1}-1}{k-1}$. This implies that the height of such a tree is $h=-1+log_k(n(k-1)+1) \leq -1+log_k(nk)=log_k n$.

Definition 5.1.5 (Universal addresses) *The address of a node in a rooted tree is a sequence. The address of the root is an empty sequence ϵ. Its children are addressed (from left to right, for example) as $1,2,3,\ldots$. When a node x in level $n \geq 1$ has children, their addresses are $x.1, x.2, x.3, \ldots$. See Figure 5.1.*

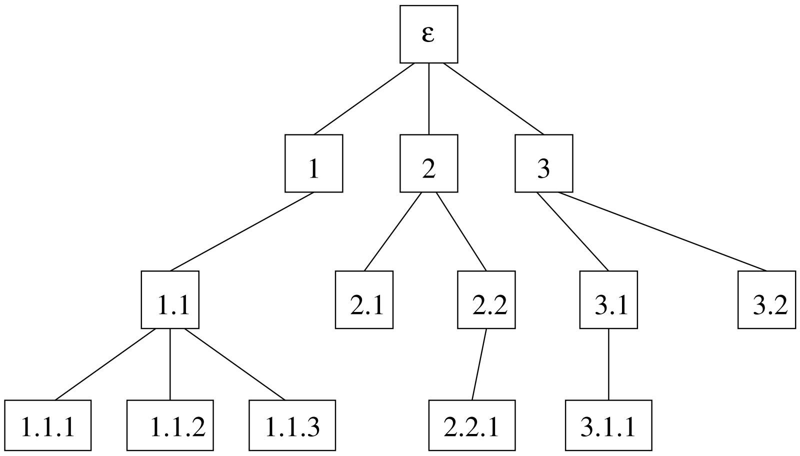


Figure 5.1: Universal addresses

Notation:

c_x is the number of x's children

h is the height of the tree

5.2 AT-GDH Protocol Description

This protocol is a generalisation of the Diffie-Hellman key-exchange to multiple participants, as are the GDH protocols, for example. It functions in a network of arbitrary topology. We call this protocol AT-GDH, where AT stands for Arbitrary Topology and GDH for Generalisation of Diffie-Hellman.

5.2.1 Construction of the Spanning Tree

Our key agreement protocol functions in an arbitrary rooted tree structure. For this purpose, a spanning tree over the graph has to be constructed first. This can be done in several ways. Below we will describe one possible protocol for constructing a spanning tree where the node initiating the protocol becomes the root. The tree is indexed using universal addresses.

In the initial state it is assumed that the nodes know their neighbours and that all links are two-way. The initiator sends a message to each of its neighbours. It thereby becomes the root of the tree and the neighbours become its children. After receiving a message, a node acknowledges it and sends a similar message to all its neighbours, except to the parent. The nodes that acknowledge a message from a node become its children in the tree. If a node gets more than one of these messages, it acknowledges and processes only the message that it receives first. Consequent messages are ignored. This continues until every node has received this kind of a message. A leaf is a node that does not receive acknowledgements from any of its neighbours. The spanning tree has now been constructed as in Figure 5.2.

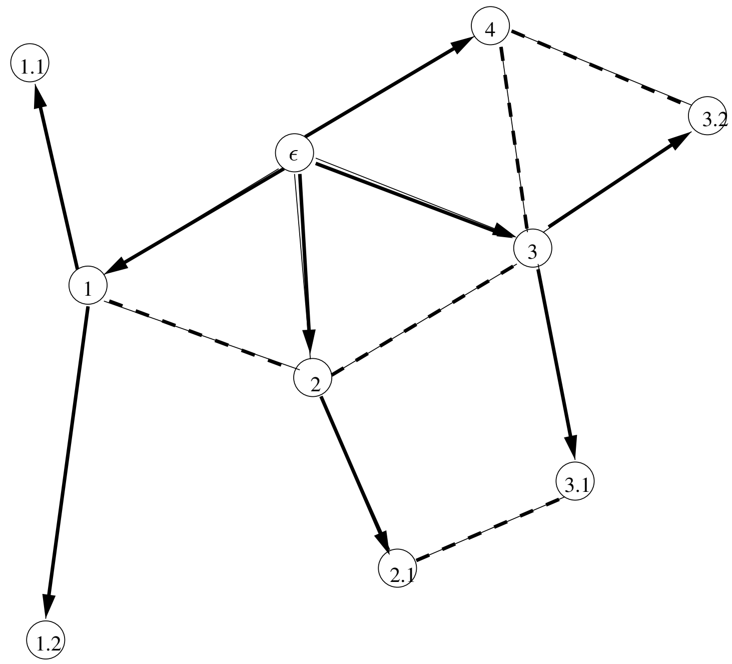


Figure 5.2: Constructing a spanning tree

This spanning-tree protocol can be used in a network where the connections are reliable. In an unreliable network, negative acknowledgements must be sent for the messages that are ignored. Local broadcast can be used to send the message to all neighbours and an acknowledgement to the parent all in one message.

5.2.2 Key Agreement in an Arbitrary Tree

In the first phase, each node's contributions to the key are gathered. The protocol progresses from the leaves to the root. The leaves start by selecting a random secret exponent e_x and calculating α^{e_x}, after which they send the result to their respective parents.

On round 2, the nodes with only leaves as children select their exponents likewise. They form Diffie-Hellman keys with their leftmost children using these exponents in the usual way but do not send their public keys to the children yet. After this half of a DH exchange, each node continues by using this newly formed DH key as a new exponent and forms a DH key with the second child. The node repeats this for all its children. The secret formed with the last child serves as the node's new private key, to which the node raises α. The resulting public key is sent to this node's parent. When the parent has received similar messages from all its children, it can repeat the same computation. This continues until the root has received all of its children's public keys. The root repeats the same kind of computation as all the other branch nodes. The secret key formed between the root and its last child will be the shared session key material for the entire network.

The reason for having each node to do these exponentiations is to have done as much of the computation as possible before sending the key shares upwards and thereby reduce the amount of total computation.

In the second phase, the root starts the distribution of the public keys down the tree. It sends the child i a message that contains $K(\epsilon, i-1)$, in which the recursive exponentiation of the key shares of the child's left-hand-side siblings has been performed, and the public keys of the right-hand-side siblings. The idea is here as well that the parent performs computations on behalf of the children as far as possible and then passes the information to the children. This also decreases the total amount of computation done during the protocol.

The message which a node x sends to its child i contains M_x, the message x itself has received from its parent, and $K(x, i-1)$, which combines the key shares of the left-hand-side siblings, and the public keys of the right-hand-side siblings.

The reason for delaying the sending of the key shares to the children is to minimize the number of messages sent during the protocol. Because the child needs to know the public keys (or the results of the recursive exponentiations of them) of all the participants, it is not efficient to send partial information down the tree. In this protocol, every node sends one message to its parent and receives one message from its parent.

Key Agreement Protocol for an Arbitrary Tree:

Initialisation: Let G be a finite cyclic group of order q, and let α be a generator of G. The participants are assumed to pick their secret exponents randomly from $\mathbb{Z}_q$. It is also assumed that there exists a bijection $\varphi : G \longrightarrow \mathbb{Z}_q$. Here the participants are identified with their universal address in the tree.

Phase 1

Round 1 For all nodes $x = y.i$ with $c_x = 0$

1. x selects a random $k_x \in \mathbb{Z}_q$
2. $x \rightarrow y : \alpha^{k_x}$

Rounds 2 ... h For all nodes x with $c_x \neq 0$

1. x selects a random $e_x \in G$
2. x waits to receive $\alpha^{k_{x.j}}$ for all $j = 1, \ldots, c_x$
3. x calculates $k_x = \varphi(K(x, c_x))$ from

$$\begin{aligned} K(x,0) &= e_x \\ K(x,j) &= \alpha^{k_{x.j}\varphi(K(x,j-1))} \text{ for } j = 1,\ldots,c_x \end{aligned}$$

4. $x \rightarrow y : \alpha^{k_x}$

Phase 2

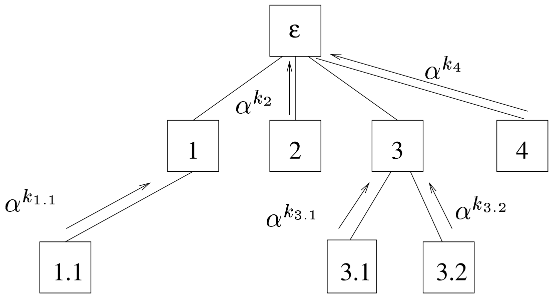
Rounds $h + l$ $l = 1, \ldots, h$ For every node $x.i$ on level l, $x \rightarrow x.i$: $M_{x.i}$, where

$$M_{x.i} = \langle M_x, \alpha^{\varphi(K(x,i-1))}, \alpha^{k_{x.(i+1)}}, \alpha^{k_{x.(i+2)}}, \alpha^{k_{x.c_x}} \rangle$$

with M_ϵ being empty.

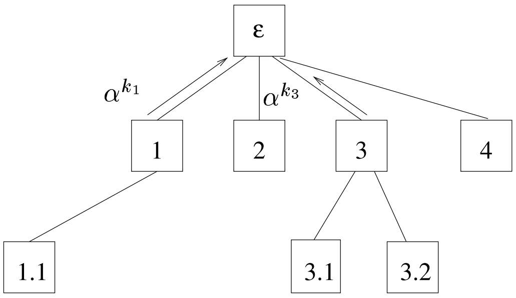
The resulting common key is $K(\epsilon, c_\epsilon) = k_\epsilon$

Example This is an example run of the protocol from Node 3's point of view: In the first round, all leaf nodes send their contribution ($\alpha^{e_{x.i}}$) to their parents. Node 3 receives the public keys of its children Nodes 3.1 and 3.2 (Figure 5.3). Then Node 3 calculates its public key α^{k_3} and sends it to its parent, Node ϵ, as can be seen in Figure 5.4. Next, the root calculates the key and sends to Node 3 the key-pieces which Node 3 needs in order to construct the whole key and to redistribute the key downwards (Figure 5.5). Finally, Node 3 sends to its children the respective messages which they need in order to construct the whole key (Figure 5.6).



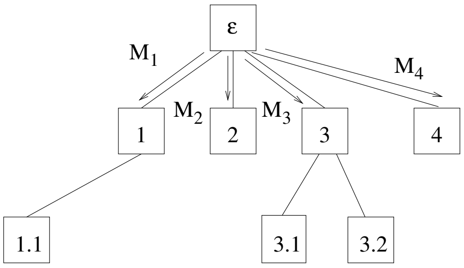
$$\begin{aligned} 3.1 &\rightarrow 3 : \alpha^{k_{3.1}} \\ 3.2 &\rightarrow 3 : \alpha^{k_{3.2}} \end{aligned}$$

Figure 5.3: Example run, step 1



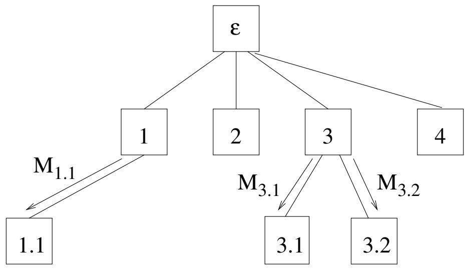
$$\begin{aligned} K(3,0) &= e_3 \\ K(3,1) &= \alpha^{k_{3.1}\varphi(K(3,0))} \\ K(3,2) &= \alpha^{k_{3.2}\varphi(K(3,1))} \\ k_3 &= \varphi(K(3,2)) \\ 3 \rightarrow \epsilon &: \alpha^{k_3} \end{aligned}$$

Figure 5.4: Example run, step 2



$$\begin{aligned} \epsilon \rightarrow 3 &: M_3 \\ M_3 &= \langle -, \alpha^{\varphi(K(\epsilon,2))}, \alpha^{k_4} \rangle \end{aligned}$$

Figure 5.5: Example run, step 3



$$\begin{array}{ll} 3 \rightarrow 3.1 & : M_{3.1} \\ M_{3.1} & = \langle M_3, \alpha^{\varphi(e_3)}, \alpha^{k_{3.2}} \rangle \\ 3 \rightarrow 3.2 & : M_{3.2} \\ M_{3.2} & = \{ M_3, \alpha^{\varphi(K(3,1))} \} \end{array}$$

Figure 5.6: Example run, step 4

Now, Node 3 knows k_3 and has received $\alpha^{\varphi(K(\epsilon,2))}$ and α^{k_4}. It can calculate the key $k_\epsilon = K(\epsilon, 4)$ from these pieces in the following way.

$$\begin{aligned} K(\epsilon,3) &= (\alpha^{\varphi(K(\epsilon,2))})^{k_3} \\ K(\epsilon,4) &= (\alpha^{k_4})^{\varphi(K(\epsilon,3))} = k_\epsilon \end{aligned}$$

5.3 Evaluation of the Arbitrary Topology Protocol

In this section, the protocol is evaluated in the light of the criteria defined in Chapter 3, i.e., efficiency, security and suitability for ad-hoc networks.

5.3.1 Security

Secrecy This protocol is a generalisation of the Diffie-Hellman key exchange to more than two participants. We repeatedly use the previously constructed key as the new exponent. The combining of the key shares follows the structure of the tree, proceeding from the leaves to the root and, at each branch node, from the leftmost child to the rightmost child. Many protocols from the literature construct the key by combining it from two participant DH-keys, but no formal proofs of security for this particular problem has been published. Becker and Wille presented a proof sketch for the Hypercube protocol, which reduces the security of that protocol to the DDH problem. It shows that if the result of the Diffie-Hellman key exchange is polynomially indistinguishable from a random number, then the key produced by Hypercube is also indistinguishable from a random number. On the other hand, it is easily seen, that if Diffie-Hellman key exchange can be broken, these protocols fail as well.

Interestingly, Steer et al. [SSDW88] use the key structure

$$K = \alpha^{(x_n \alpha^{(x_{n-1})\alpha^{(\ldots\alpha^{(x_1 x_2)})}})}$$

in a star topology. This is the same structure that is used in our protocol to combine the leaf nodes' key parts in the first step of the protocol. The security of this protocol has not been formally proven. The proof of security for our protocol would imply the security of this protocol. We conjecture that the security of our protocol can be reduced to the DDH problem. While no proof of this is available at this time, we anticipate that techniques similar to those used by Becker and Wille can lead to a formal proof.

Other security properties The *operationality* property, which requires that the participants know the identities of other participants that are present in the protocol can be fulfilled by collecting the information along the way together with the key-pieces. However, this is unnecessary if the participants are satisfied with the knowledge that everyone is a member of the group. An *Unused fresh key* is guaranteed by the use of Diffie-Hellman key exchange that is *contributory.* This also guarantees the *lack of key control*, since every participant can independently choose a fresh key element.

In order to calculate the key, knowledge of at least one secret exponent is required, assuming that the generalisation is as secure as the DH key exchange. As an honest participant is not supposed to reveal her key element, outsider can learn the key only with the help of a dishonest participant. This provides *implicit key authentication.*

Key confirmation might be realised by acknowledgement messages after the completion of the protocol. However, it would require too many all-to-all messages, which may not even be possible for reasons explained in Chapter 3. Therefore key confirmation and *explicit key authentication* are not considered in this protocol.

Resilience: If authentication is implemented, an outsider should not be able to mislead the participants in any other way but a denial of service (DoS) attack. On the other hand, a dishonest participant can create more problems with a DoS attack. An unavoidable consequence of using the efficient tree structure is that the child nodes must trust their parent nodes against DoS attacks. A parent node could change the key-part before sending it down the tree or refuse to send (or not be able to send) anything downwards altogether. However, similar problems are common in other protocols as well and cannot be solved without extensive losses in efficiency. It should be noted that in this protocol, the effect of the DoS attack in the middle of the protocol run is limited to the children of the untrustworthy or removed node. We have assumed that the nodes know who their neighbours are. Reliable implementation of the neighbour discovery is, of course, a separate problem.

As the created key is fresh, it is independent of any former keys and possible long term secrets. Therefore the protocol is *resistant to known key attacks.* This property results from the use of one-time public keys in the Diffie-Hellman exchange. The protocol itself does not use any long term secrets and does not affect their secrecy.

5.3.2 Efficiency

This section deals with the communication complexity of the protocol. As the topology of the spanning tree is arbitrary, i.e. the number of children and the height of the branches is not limited, exact figures are not always possible. When this is the case, we estimate the figures by assuming that the tree is a balanced perfect k-ary tree. We have marked the values that are estimated this way with an asterisk in Tables 5.1 and 5.3.

Messages In the first phase, every node except the root, sends one message. This makes the number of messages in the first phase $n-1$. In the second phase, every node except the root receives one message. This makes the total number of messages $2n-2$, which is optimal when broadcast is not allowed.

Exchanges This protocol can be seen so that every non-root node makes a Diffie-Hellman key exchange with its parent. When calculated this way the number of exchanges equals $n-1$, which seems to be even better than the optimal values given in Chapter 3. This discrepancy is due to the fact that most of the exchanges are not simultaneous but the child node may have to wait for the response for several rounds. In the worst case, a leaf sends its contribution at the beginning of the protocol and receives an answer at the end of the protocol. Therefore, the number of exchanges in this protocol is not actually comparable to that of the other protocols. If every message is counted as one exchange, the number of exchanges is $2n-2$, which is still close to optimal when broadcast is not allowed.

Simple rounds When calculating simple rounds, we assume that every node can send and receive at most one message per round. It is not possible to give exact figures for the number of simple and synchronous rounds since the actual topology is not fixed. Instead, we use estimates for the average and worst case scenarios. The number of simple rounds depends on the number of children an average node has and on the height of the tree. Since every node can receive just one message per round, a parent node with k children has to wait k rounds before it can send its own message. For example, in a perfect balanced k-ary tree, the number of children is k and the height of the tree is less than or equal to $\log_k n$. Hence, the number of simple rounds is less than $2k \log_k n$. The worst case scenario is when all nodes have to send their contribution sequentially. This could happen in a star topology ($k = n-1$) or when the nodes form a chain ($k = 1$). So, in the worst case, the number of simple rounds equals $2(n-1)$. This situation is, however, unlikely in a random arbitrary network.

Synchronous rounds In a synchronous round, a node can send or receive many messages in one round. In the first phase, we assume that a parent node receives a message from all of its children simultaneously in one round and sends its own message in the next. In the second phase, a parent node sends the necessary key information to all of its children in one round. This makes the total number of synchronous rounds equal to 2 times the height of the tree, which is usually logarithmic to the number of nodes n. The worst case is again when the nodes form a chain.

Local Computation When the protocol is in Phase 1, i.e. the messages are travelling upwards, every node performs as many exponentiations as it has children, plus one. When the protocol is in Phase 2, i.e. messages are travelling downwards, the situation is more complicated. The number of exponentiations a node has to perform is $l+N$, where l equals the level of the node in the tree, i.e. the number of ancestors the node has, and N equals the number of the ancestors' right-hand side siblings plus the right-hand side siblings of the node itself.

In a balanced perfect k-ary tree where all leaves are at the same level, the leftmost node one level up from the bottom level ($l = h-1$), will do most exponentiations: $k+1$ in Phase 1 and $l+l(k-1) = lk = k(h-1)$ in Phase 2. The sum is $kh+1 \leq k \log_k n+1$ This is the worst case, though. The right-most sibling on level one will perform only one exponentiation in phase 2. The exact figure for any node of a perfect k-ary tree can be calculated by using the universal addresses: A node $(x_1.x_2.\ldots.x_l)$ will perform $k+1+l+\sum_{i=1}^{l}(k-x_i)$ exponentiations.

Total Computation This value is also estimated by using a balanced perfect k-ary tree, where all leaves are at the same level. The total number of exponentiations performed is calculated by summing over all nodes in the tree: $\sum_{allnodes}(k+1+l+\sum_{i=1}^{l}(k-x_i)) = \mathcal{O}(n \log_k n)$.

The efficiency comparison does not do justice to our protocol in all respects. This is due to the fact that, in the other solutions, it is assumed that it is always possible to arrange the nodes according to the required topology. In a typical ad-hoc network, we do not have this kind of flexibility in choosing an optimal topology for the protocol. Instead, the protocol has to adapt to

messages	$2n-2$
exchanges	$2n-2$
simple rounds	$2k\lceil\log_k n\rceil^*$
syn. rounds	$2\lceil\log_k n\rceil^*$
broadcast	0
local computation	$k\lceil\log_k n\rceil+1^*$
total computation	$\mathcal{O}(n\log_k n)^*$

Table 5.1: The efficiency parameters for the AT-GDH protocol

whatever topology the actual communication links happen to form. This will obviously affect the efficiency of the protocol.

	lower bound	GDH.2
messages	n	n
exchanges	n	n
simple rounds	$\lceil\log_2 n\rceil$	n
syn. rounds	1	n
bc	–	1
local exp.	–	$n+1$
total exp.	–	$\mathcal{O}(n^2)$

Table 5.2: The efficiency measures for protocols that use broadcast

	lower bound	Hypercube	Octopus	AT-GDH
messages	$2n-2$	$n\lceil\log_2 n\rceil$	$3n-4$	$2n-2$
exchanges	$2n-4$	$\lceil\frac{n\log_2 n}{2}\rceil$	$2n-4$	$2n-2$
simple rounds	$\lceil\log_2 n\rceil$	$\lceil\log_2 n\rceil$	$2\lceil\frac{n-4}{4}\rceil+2$	$2k\lceil\log_k n\rceil^*$
syn. rounds	–	$\lceil\log_2 n\rceil$	4	$2\lceil\log_k n\rceil^*$
local exp.	–	$1+\lceil\log_2 n\rceil$	$2\lceil\frac{n-4}{4}\rceil+5$	$k\lceil\log_k n\rceil+1^*$
total exp.	–	$\mathcal{O}(n\log_2 n)$	$\mathcal{O}(n)$	$\mathcal{O}(n\log_k n)^*$

Table 5.3: The efficiency measures for protocols without broadcast

5.4 Using AT-GDH in Ad-hoc Networks, Practical Considerations

The purpose of the protocol is to create a session key material for encrypting and authenticating communication within the group. The resulting $K_(\epsilon, c_\epsilon)$ is never used in encrypting messages as such, instead we would use a one directed hashing function, something like $hash(\text{“encryptionkey}'', k_\epsilon)$, to produce the key.

In order to add a partial key confirmation, the information about the structure of the tree or at least information of the identities of all participants could be added to the Phase 2 messages. It would not increase the number of messages, because this information can be gathered on the way, but it would increase the length of the messages.

The spanning tree structure is a temporary construction meant only for the purposes of the key agreement. After the protocol is completed, the tree can be discarded. It is not reasonable to change the structure of the key according to group movements if the group management does not require it. The spanning tree may but does not necessarily have to follow any military or other sort of hierarchical structure which the group might have.

5.5 Future Work

AT-GDH is meant for initial key agreement when the participants do not yet share any secret keys. After completion of the protocol, it is possible, and, in ad-hoc networks, even likely that the network structure changes. It may become necessary to add a new member to the group, remove someone, split the group or merge two groups. Instead of creating a new key each time, it would be convenient to change the existing key structure as little as possible. This is called auxiliary key agreement (AKA) and there are several choices for adding this kind of dynamicity to the protocol. A member can be removed, for example, by forming the new key from the key shares of the remaining members. This requires a partial re-execution of Phase 2. The auxiliary key agreement is a probable next step in the development of the key agreement protocol.

It is possible, for example in radio connections, that a node can make a local broadcast, i.e., broadcast a message to all of its neighbours simultaneously. Taking advantage of local broadcast could be done in the second phase simply by merging the messages sent for children and removing the redundant information. This would reduce the number of messages in the second round to $\log_k n$, and therefore increase the efficiency of the key agreement.

The amount of total computation done seems to depend on the shape of the spanning tree. The key structure is not symmetrical in the tree. A node receives the left-hand part of the tree as readily calculated, but it has to exponentiate the right-hand side pieces itself. Therefore, it may be possible that by adjusting the shape of the spanning tree (when the network topology allows it) the amount of total computation in the protocol could be optimised. This possibility needs to be further investigated.

It is generally a good idea to update the encryption key once in a while. A straightforward way to accomplish key update is to have one participant change her key share and distribute it to everyone. Other possibilities should also be looked into.

This protocol (as most of the other protocols in the literature) is based on the Diffie-Hellman key exchange. Another possibility we have considered is a nonce-based protocol, i.e., changing the key-shares by nonces generated by each participant and using a hash value of their combination as the key material. This might speed up the calculations.

Chapter 6

Conclusions

The purpose of this thesis was to find an efficient and secure key establishment protocol suitable especially for ad-hoc networks. The existing protocols for group key-establishment were overviewed and compared in light of the special efficiency, security and operational requirements of ad-hoc networks. The focus was on the protocols from the literature that are contributory, extensions of the Diffie-Hellman key exchange to multiple participants: GDH.2, Hypercube and Octopus. When evaluating efficiency we measured the data transfer requirements, time complexity and amount of computation in the protocols. We also paid attention to properties that ensure security in a situation where no previous secrets exist and the participants do not trust each other completely. The security of the protocols was considered in the passive adversary model as is the case with the original Diffie-Hellman key exchange.

We found that there are many protocols with good efficiency and security properties but, unfortunately, none of the protocols from the literature is suitable for all types of ad-hoc networks. This is mainly because they demand the network topology to follow a predestined structure, such as a chain in GDH.2 or a "multi-dimensional" cube as in Hypercube. Therefore we presented an alternative, a new protocol in a tree-topology. It operates in a network of arbitrary topology, which comes from the fact that a spanning tree can be constructed in any connected graph.

Our new protocol was found to have very good data transfer capacity figures, even optimal in the number of messages. The time complexity properties were good as well, the second fastest in a comparison measured in synchronous rounds. In the amount of computation, AT-GDH requires the smallest number of local exponentiations and the number of global exponentiations is among the smallest of the surveyed protocols.

AT-GDH is meant for the initial key agreement in an ad-hoc group. It serves as a good basis that can be extended to fully dynamic key agreement with the few additions of auxiliary key agreement procedures that are needed when group memberships change. Augmented with a good authentication method it can be made resistant to active attacks as well.

As future work, we suggested adding auxiliary key agreement, key update and taking advantage of local broadcast. The possibilities of speeding up the protocol by adjusting the shape of the spanning tree and by using nonces instead of Diffie-Hellman key shares also need to be further investigated.

Bibliography

[AG00] N. Asokan and Philip Ginzboorg. Key-agreement in ad-hoc networks. *Elsevier Preprint*, 2000. To appear.

[AST98] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. Authenticated group key agreement and friends. In *Proc. 5th ACM Conference on Computer and Communications Security*, pages 17–26, San Francisco, CA USA, November 1998. ACM Press.

[AST00] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. New multiparty authentication services and key agreement protocols. *IEEE Journal on Selected Areas in Communications*, 18(4):628–640, April 2000.

[AT99a] Giuseppe Ateniese and Gene Tsudik. Group signatures á la carte. In *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 848–849, Baltimore, MD USA, January 1999. ACM Press. USC Technical Report 98-676.

[AT99b] Giuseppe Ateniese and Gene Tsudik. Some open issues and new directions in group signatures. In *Proc. 3rd International Conference on Financial Cryptography (FC'99)*, volume 1648 of *LNCS*, pages 196–211, Anguilla, British West Indies, February 1999. Springer.

[BD94] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *LNCS*, pages 275–286, Perugia, Italy, May 1994. Springer.

[BM92] Steven M. Bellovin and Michael Merrit. Encrypted key exchange: password-based protocols secure against dictionary attacks. In

Proc. 1992 IEEE Symposium on Research in Security and Privacy, pages 72–84, Oakland, CA USA, May 1992. IEEE Computer Society Press.

[Bon98] D. Boneh. The decision diffie-hellman problem. In *Proc. of the Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*, pages 48–63, Portland, Oregon, USA, June 1998. Springer.

[BW98] Klaus Becker and Uta Wille. Communication complexity of group key distribution. In *Proc. 5th ACM Conference on Computer and Communications Security*, pages 1–6, San Francisco, CA USA, November 1998. ACM Press.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[Die97] Reinhard Diestel. *Graph theory.* Springer-Verlag, New York, 1997.

[ITW82] Ingemar Ingemarsson, Donald T. Tang, and C. K. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, IT-28(5):714–720, September 1982.

[KPT00] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *ACM Conference on Computer and Communications Security*, pages 235–244, 2000.

[MAH00] Silja Mäki, Tuomas Aura, and Maarit Hietalahti. Robust membership management for ad-hoc groups. In *Proc. 5th Nordic Workshop on Secure IT Systems (NORDSEC 2000)*, Reykjavik, Iceland, October 2000.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[SSDW88] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A secure audio teleconferencing system. In *Advances in Cryptology -CRYPTO '88*, volume 403 of *LNCS*, pages 520–528, Santa Barbara, CA, USA, August 1988. Springer.

[STW96] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie"-hellman key distribution extended to group communication. In *3rd ACM Conference on Computer and Communications Security*, pages 31–37, New Delhi, India, March 1996. ACM Press.

[STW98] Michael Steiner, Gene Tsudik, and Michael Waidner. CLIQUES: a new approach to group key agreement. In *Proc. 18th International Conference on Distributed Computing Systems (ICDCS'98)*, pages 380–387, Amsterdam, The Netherlands, May 1998. IEEE Computer Society Press.

[STW00] Michael Steiner, Gene Tsudik, and Michael Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(08), August 2000.