

Rheinisch-Westfälische Technische Hochschule Aachen

Lehrstuhl für Informatik IV
Prof. Dr. rer. nat. Otto Spaniol



Establishing Pair-wise Keys for Secure Communication in Ad-Hoc Networks

Seminar: Pervasive Computing
SS 2004

Urs Enke
Matrikelnummer: 227450

Betreuung: Ralf Wienzek
Lehrstuhl für Informatik IV, RWTH Aachen

In so-called ad-hoc networks, nodes arbitrarily join and leave, creating a dynamic topology requiring the participants to repeatedly create secret keys for their mutual communication. In the following, a protocol is presented that enables the participating devices to establish such keys with respect to a trade-off between communicational cost and the probability that colluding nodes in the network gain knowledge of others' keys.

Contents

1	Introduction	4
1.1	Ad-hoc Networks	4
1.2	Security Considerations	4
2	Ideas and Related Work	5
2.1	Probabilistic Key Sharing	5
2.2	Shared Secrets	6
3	The Ad-hoc Key Establishment Protocol	7
3.1	Assumptions	7
3.2	Key Pre-Distribution	7
3.3	Pair-wise Key Establishment	8
4	Security Parameters	11
5	Complexity and Performance	13
5.1	Computation	13
5.2	Communication	14
5.3	Trade-Offs	14
5.3.1	Experiment I	14
5.3.2	Experiment II	15
5.3.3	Experiment III	15
5.3.4	Experiment IV	16
6	Failure and Corruption of Nodes	17
6.1	C_2 Proxy Corruption	17
6.2	C_3 Proxy Corruption	18
7	Summary	19

1 Introduction

1.1 Ad-hoc Networks

Many communication applications are not possible with wired connections between participants, so in some of these cases, only comparably few infrastructural nodes, with which single mobile ones communicate wirelessly, are connected by wire. Important examples include mobile phone networks or wireless LAN. Some applications, though, do not practically allow the set-up of such a wired backbone network, or even any kind of centralized administration or established infrastructure, be it wired or wireless. They rely entirely on communication between the participating nodes, which thus have to be able to dynamically set up such a decentralized network ‘on the spot’, hence called an *ad-hoc network*.

The self-organisational aspect is further emphasized by the demand for flexibility beyond the initiating phase: Single nodes may arbitrarily leave or join the network, so that its current topology can be very dynamic. A simple example of an ad-hoc network (that actually does not necessitate the ad-hoc model) could be that formed in a business meeting by the participants’ notebooks. In a more complex variant, rescue workers could set up a network on-site in order to coordinate their efforts.

1.2 Security Considerations

A way to establish a secure communication link between two nodes is to share a secret key between them. The limited storage capacity of certain kinds of mobile devices already rules out the naive idea of pre-loading each one with separate keys for all others (for all n nodes together, these would be $O(n^2)$ keys), not to mention the problematic definition of ‘all others’, as it is in ad-hoc networks’ nature to require communication with nodes never encountered before.

Public-key systems may not be a viable solution due to their infrastructural requirements and the complex calculations involved, which may impose too long delays on the potentially computationally slow nodes. Also, complex operations contribute to the quicker exhaustion of the potentially very limited energy resources. Key-distribution centres could distribute symmetric keys, yet their existence contradicts ad-hoc networks’ requirement to be robust against the sudden departure (or failure) of participating nodes.

Especially due to the latter aspect two nodes should determine a new secret key that can be used to encrypt their messages each time a session is to be established, or at least if a certain time has elapsed since their last communication. Obviously, the major problem with this approach is the key establishment itself that needs to take place over insecure communication links. In [1], Zhu et al. propose a solution to this problem that requires a central key server only at initialisation time,¹ and makes the required storage capacity per node only dependent

¹A limitation concerning this aspect will be mentioned in section 3.2.

on the intended level of security, i.e. the probability that other nodes do actually know the key established by the nodes in question.

2 Ideas and Related Work

What shall henceforth be called the *Ad-hoc Key Establishment (AKE)* protocol starts off with a phase in which all participating nodes are pre-loaded with pseudo-random subsets of a base set of keys. As these subsets are not necessarily disjoint, two nodes that want to establish a secret key for their mutual communication choose from different *logical paths*² through the network to organize this: If node u shares a pre-loaded key with node x , and so does x with v , then u can communicate with v via x . In this setting, x has to decrypt and re-encrypt the respective messages.³

The AKE protocol's basic idea now is to let u split its key-establishment messages into several parts and transmit each one on a different logical path to the destination node v : Obviously, the intermediate nodes can perfectly well read the partial data sent via them, and so can other nodes that happen to receive transmissions encoded with keys they possess. However, only a coalition amongst them of a certain minimal size can also determine the (whole) secret key u sent to v as basis for their future mutual communication.

Before describing the protocol in far more detail in the next section, it is both interesting and sensible to take a quick glance at the two separate notions that it is based on.

2.1 Probabilistic Key Sharing

Approaches to base communication on not necessarily disjoint sets of keys are known under the name of *set intersection schemes*: Here, two nodes intending to communicate can encrypt their messages using all the keys they share, and the system has to be set up in a way that no other node will know all the keys in that intersection [2].

An extension to this is to relax the latter demand and only require that the probability for some other node being able to decrypt the messages is very low. For example, in [3], Dyer et al. show bounds on the total number k of keys needed for n participating nodes: If for each node, every key is considered and included in the node's set of keys with probability $p = \frac{2}{3}$ (which proves optimal in this context), $k = 13 \cdot \log_2 n$ suffices to have a probability of more than 0.5 that no triple of nodes can be found in which one node's keys contain all those that the other two share. In other words, with these parameters, it is less likely that some node can decrypt two others' direct communication (provided that they make use of all their mutual keys) than

²This term will be explained later; at this stage it suffices to say that several logical paths may be located on the same physical one.

³Although the above-mentioned 'sets of keys' that the nodes are pre-loaded with are also secret in the sense that only part of the nodes know them, the use of the term *secret key* will henceforth, in line with cryptographic jargon, exclusively refer to the mutually shared key that the nodes u and v try to establish.

that none of them can. Note that in this scheme, the number of keys per node will vary. In case a fixed-sized set of keys is assigned to each node, the number of required keys can be reduced further.

Dyer et al. also describe a de-randomized version of their algorithm, but the above still serves as a good example for probabilistic key sharing.

2.2 Shared Secrets

What the AKE protocol adds to the above key sharing methods is not to use one combined key but instead to encrypt parts of the secret message with subsets of keys. As originally proposed by Shamir in [4] and to be mentioned again in section 6.2, this distribution can take place redundantly over various shares (which are not simply partial strings) and thus give a certain robustness against the failure of transmission links.

A more traditional cryptographic use of shared secrets has rather been seen in the possibility to share a key among a number of parties of which a subset of a certain size is needed to reconstruct the shared key, for example to sign a contract. More ingenious protocols even provide that the common key is not disclosed to the participants during the signing procedure, so that further misuse is prevented: In an example where the shared key is a company's signature and the participants are its directors, a majority could sign a contract, yet no director could copy that signature for use with other documents.

A simple example for a protocol that merely serves the reconstruction of a shared secret is the one Shamir suggested: Let $S \in \mathbb{N}$ be the secret, m the number of participants and r the latter's number required to cooperate in order to determine the secret.

- The secret's original owner, called *server*, chooses a prime number $p > \max(m, S)$.
- The server randomly chooses a polynomial $f(x) = [a_0 + a_1x + \dots + a_{r-1}x^{r-1}]_p$, where the $a_i \in \mathbb{Z}_p$ are uniformly distributed and $f(0) = a_0 = S$.
- Each of the participants i gets a value $f(i)$.
- Whoever is to receive the secret asks r participants for their function values.
- The value pairs $(i, f(i))$ are used to interpolate the polynomial $f(x)$.
- The secret can be computed as the value of $f(0)$.

Example 2.1 Let $S = 2$, $m = 5$ and $r = 2$.

- $p := 13$
- The server chooses $a_1 = 3 \implies f(x) := [3x + 2]_{13}$

- | | | | | | |
|--------|---|---|----|---|---|
| x | 1 | 2 | 3 | 4 | 5 |
| $f(x)$ | 5 | 8 | 11 | 1 | 4 |

- *The participants 2 and 4's function values are used for reconstruction:*
 $f(2) = 8 \wedge f(4) = 1 \implies f(x) = [3x + 2]_{13}$
- $S = f(0) = 2$

This method works because interpolating a polynomial of degree $r-1$ requires at least r function values (at distinct locations). As soon as even only one of them is missing, the value $f(0)$ is not only not known, but also all elements of \mathbb{Z}_p have the same probability to be that value, so no information can be drawn from an incomplete coalition.

3 The Ad-hoc Key Establishment Protocol

3.1 Assumptions

Before describing the AKE protocol proposed by Zhu et al., the conditions that it requires to work need to be mentioned.⁴ These are basically two:

- The underlying network's connections are symmetric, i.e. all nodes can transmit into all directions and have equal power to do so.
- At initialisation time (but not later), a central key server is needed to initialize the participating nodes.

The protocol's central aim is to make certain coalitions of compromised nodes powerless concerning the uncovering of secrets sent via them. It should thus be stressed that nodes' classification as *compromised* means that all their information (plus any acquired by eavesdropping) can be assumed publicized to an attacker, i.e. that there is no practical distinction between the set of colluding nodes and 'the attacker'.

3.2 Key Pre-Distribution

In section 2, a sketchy description of the AKE protocol was already given. In the following, this is taken to a more formal level. As repeated above, the initial phase comprises the distribution of auxiliary keys to the participating nodes:

Server: Define a pool $P := \{k_1, k_2, \dots, k_l\}$ of l keys, each randomly chosen from \mathbb{Z}_p .
 To each node u assign a pseudo-random set of keys $\{k_{i_1^u}, \dots, k_{i_m^u}\} =: R_u \subset P$.

⁴Apart, of course, from the motivational supposition that energy, power and computational resources within the nodes are limited.

The algorithm that the key server uses to assign the keys to the participating nodes is meant to be a publicly known, deterministic mapping from the node’s ID u to the indices i_1^u, \dots, i_m^u that define the set R_u . This way, each node can, for itself, determine the indices of any other node’s keys and thus know which ones they share.

Note that this procedure either requires all nodes that may ever want to establish a pair-wise key to be present before the actual networking begins, or necessitates that the key server be on stand-by to serve any newcomers that did not participate in the key pre-distribution. In a way, this contradicts the protocol’s purpose to be used in ad-hoc networks (where nodes can spontaneously become participants), or at least its authors’ claim that the server only needs to be active at initialisation time.

3.3 Pair-wise Key Establishment

The following describes the steps taken to actually establish a secret key. As mentioned in section 2.2, shares could be created in a redundant fashion that yields robustness against their partial loss. Here, however, we only treat the simple version; a closer look at the alternative will be taken in section 6.

- Sender: Randomly generate a secret key S and bit strings s_1, \dots, s_{n-1} of the same length. Compute⁵ share $s_n := S \oplus s_1 \oplus \dots \oplus s_{n-1}$. Transmit all shares in encrypted form to the recipient, each using a separate *logical path* and a different set of keys.
- Receiver: Reconstruct the secret key from the shares as $S = \bigoplus_{i=1}^n s_i$. Send a **Hello** message to the sender, using S as the key for an enclosed authentication code.⁶
- Sender: Verify the **Hello** message. If this fails or times out, abort or re-try the transmission of shares.

Questions arise what would be a sensible choice for the number n of shares, and what those above-mentioned separate paths are that the shares are to take. Given a quantitative level of demand on security (which will be discussed later in this section), a certain number of shares will be sufficient. As these can be transmitted via different types of communication paths provided by the network topology, called *logical paths*, it seems plausible to first use paths that cause the least communication overhead and only use worse ones if required to meet the demands. The following classes of logical paths are considered by the AKE protocol:

- C_1 : In case u and v have common keys, one share can be encrypted using the key $\bigoplus_{k \in R_u \cap R_v} k$. As decoding requires knowledge of all used keys, the separate encryption of several shares using subsets of the common keys would not yield a higher level of security.

⁵ \oplus is the logical XOR.

⁶Actually, the authors propose to use S as the seed for a pseudo-random function f whose value $f(0)$ is to serve as authentication code and whose value $f(1)$ is used as the ‘real’ secret key. This way, cryptographic *known plaintext* attacks making use of the knowledge that **Hello** is transmitted, are thwarted.

C_2 : Preferably helped by information provided by the underlying routing protocol,⁷ a set of nodes can be determined that are on *physical paths* (i.e. ‘direct’ in routing terms, e.g. the shortest one) between u and v and share keys with both of them. Just as in C_1 u communicates directly with v , it can use the same XORing technique with the keys it shares with an intermediate node, called *proxy*, which then does the same with the keys it, in turn, shares with v .

Note that the AKE protocol requires that every key is only used once. Perhaps for the sake of simplicity of the security formulae that are treated in section 4, the protocol’s authors included the rule that the number of keys used in one of the two steps (u to proxy, proxy to v) should not exceed that used in the other. Which of the keys remain unused either needs to be told to the receiver of the transmission in question, or has to be standardized, e.g. by always using the keys with the lowest indices.

C_3 : If needed, proxies can be used that do not lie on any of the direct paths between u and v used for C_2 . To minimize communication overhead, AKE considers only neighbours of either u or v , as information about those could easily be added to messages exchanged by the routing protocol, anyway.

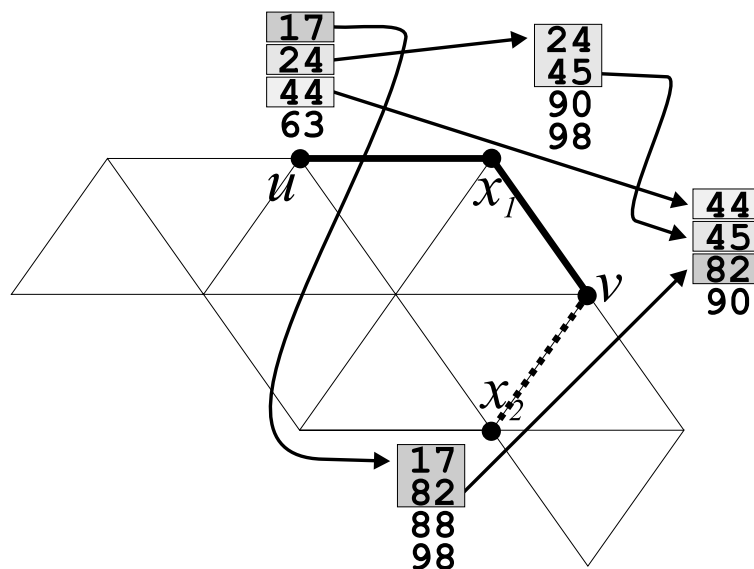


Figure 1: Example for the path classes

Figure 1 depicts a simplified example where node u can employ paths of all three classes for transmitting shares to node v :

C_1 : Directly by use of key 44.

C_2 : Indirectly, but without a detour, via x_1 , using keys 24 and 45.

⁷Zhu et al. note that this is the case with the drafted Dynamic Source Routing Protocol. [5]

C_3 : Indirectly, using a detour via x_2 , using keys 17 and 82.

The following algorithm formally describes how, given a node pair (u, v) and a security parameter p_0^w , the number of required secret shares n can be determined, and how the multiple use of keys can be prevented by managing sets of keys. It is constructive in that it immediately yields the paths that can be used, and also shows that the number of secret shares is only increased as much as needed to meet the security demands.⁸

The parameter p_0^w specifies the maximal allowed probability that an arbitrary coalition of w nodes knows all the keys used by u and v and can thus recover the pair-wise key. A formula for the actually attained probability p^w will be developed in the next section.

Let $I_{ab} := I_a \cap I_b$

0. $n := 0$

$I_u :=$ the key indices that define the set R_u

$I_v :=$ the key indices that define the set R_v

1. Determination of paths of class C_1

– $z_1 := |I_{uv}|$

– $n := \text{sgn}(z_1)$

– $I'_u := I_u \setminus I_{uv}$

$I'_v := I_v \setminus I_{uv}$

2. Determination of paths of class C_2

– $P := \{\text{proxy candidates of class } C_2\}$, $z_2 := 0$

– WHILE $P \neq \emptyset \wedge p^w(z_1, z_2) \leq p_0^w$ DO

1. Randomly choose $x \in P$

2. Let $I_1 := I_{ux} \cap I'_u$, $I_2 := I_{xv} \cap I'_v$, $z := \min(|I_1|, |I_2|)$

3. Delete z IDs $i \in I_1$ from I'_u and z IDs $j \in I_2$ from I'_v

4. $n := n + \text{sgn}(z)$, $z_2 := z_2 + z$, $P := P \setminus \{x\}$

3. Determination of paths of class C_3

– $P := \{\text{proxy candidates of class } C_3\}$

– WHILE $P \neq \emptyset \wedge p^w(z_1, z_2) \leq p_0^w$ DO

1. Randomly choose $x \in P$

2. Let $I_1 := I_{ux} \cap I'_u$, $I_2 := I_{xv} \cap I'_v$, $z := \min(|I_1|, |I_2|)$

⁸If those demands cannot be met, the algorithm could be extended so that several proxies may be used for one logical path (so-called *multi-hop* proxies), admitting an increase in communication for the sake of improved security.

3. Delete z IDs $i \in I_1$ from I'_u and z IDs $j \in I_2$ from I'_v
4. $n := n + \text{sgn}(z)$, $z_2 := z_2 + z$, $P := P \setminus \{x\}$

In the case that not enough proxies are available to satisfy the security requirement, a less secure secret key could be temporarily established and later, when additional proxies are available (e.g. through XORing) be extended to a sufficient degree of security. Note that the proxy nodes formerly used for setting up the temporary key no longer need to be present at this stage; the described technique therefore fits ad-hoc networks' dynamic nature well.

An example for the above algorithm will be shown not before the end of the following section on security parameters, in order to enable the reader to reproduce the probability calculations. Also, the algorithm is the basis of experiments described in section 5 that determine the required number of shares under certain conditions. To be able to parameterize those experiments by their achieved level of security, it is necessary to first take a look at how that level depends on the simulated environment.

4 Security Parameters

In the pre-distribution phase, each key has a probability of m/l (i.e. the number of base keys per node divided by their total number) to be chosen by any specific node. A formula for the probability p_1^w that a coalition of w nodes knows all the z_1 keys in R_{uv} can be derived as follows:

the probability that...	
a specific node does not know a specific key	$1 - \frac{m}{l}$
a coalition of w nodes does not know a specific key	$(1 - \frac{m}{l})^w$
a coalition of w nodes does know a specific key	$1 - (1 - \frac{m}{l})^w$
a coalition of w nodes knows all z_1 keys	$p_1^w(z_1) := (1 - (1 - \frac{m}{l})^w)^{z_1}$

Class C_2 can be treated similarly, the only difference being that any transmission from u to v is vulnerable both on the way to and away from the proxy node, so the exponent w needs to be doubled:

a coalition of w nodes does not know two specific keys	$(1 - \frac{m}{l})^{2w}$
a coalition of w nodes knows all the keys required to decode all of the z_2 secret shares	$p_2^w(z_2) := (1 - (1 - \frac{m}{l})^{2w})^{z_2}$

Thus, the security p_w of the pair-wise key, defined as the probability that a coalition of w nodes is able to reconstruct all secret shares, is

$$p^w(z_1, z_2) = p_1^w(z_1) \cdot p_2^w(z_2).$$

In case not all the required paths can be realized due to a shortage of keys, there is an additional limit that should be made non-critical (i.e. yield a lower probability of the coalition's success than p_0^w) by choosing m and l appropriately: p_3^w is the probability that a node's key set

is completely covered by a coalition of w nodes:

$$p_3^w = (1 - (1 - \frac{m}{l})^w)^m.$$

Example 4.1 Referring to the network structure shown in figure 1 and assuming that the underlying parameters are $l = 100$ (number of base keys), $m = 4$ (keys per node), $w = 3$ (size of expected malign coalition of nodes) and $p_0^w = 1\%$ (upper limit on the probability of the coalition's success), the following is a very simple example of the algorithm that determines the required number of shares.

0. $n := 0$

$$I_u := \{17, 24, 44, 63\}$$

$$I_v := \{44, 45, 82, 90\}$$

1. Determination of paths of class C_1

$$- z_1 := |I_{uv}| = |I_u \cap I_v| = |\{44\}| = 1$$

$$- n := \text{sgn}(z_1) = 1$$

$$- I'_u := I_u \setminus I_{uv} = \{17, 24, 63\}$$

$$I'_v := I_v \setminus I_{uv} = \{45, 82, 90\}$$

2. Determination of paths of class C_2

$$- P := \{\text{proxy candidates of class } C_2\} = \{x_1\}, z_2 := 0$$

$$- \text{WHILE } P \neq \emptyset \wedge p^w(z_1, z_2) \leq p_0^w \text{ DO}$$

($p^w(1, 0) \approx 11.5\% > 1\% = p_0^w$, so at least one loop is necessary.)

1. Only one candidate for $x \in P$, so $x := x_1$

2. Let $I_1 := I_{ux} \cap I'_u = \{24\}$, $I_2 := I_{xv} \cap I'_v = \{45, 90\}$, $z := \min(|I_1|, |I_2|) = 1$

3. Delete IDs from I'_u and I'_v : $I'_u := \{17, 63\}$, $I'_v := \{82, 90\}$

4. $n := n + \text{sgn}(z) = 2$, $z_2 := z_2 + z = 1$, $P := P \setminus \{x\} = \emptyset$ (no more loops possible)

3. Determination of paths of class C_3

$$- P := \{\text{proxy candidates of class } C_3\} = \{\dots, x_2, \dots\}$$

$$- \text{WHILE } P \neq \emptyset \wedge p^w(z_1, z_2) \leq p_0^w \text{ DO}$$

($p^w(1, 1) \approx 2.5\% > 1\% = p_0^w$, so one loop is necessary.)

1. Randomly choose $x \in P$: $x := x_2$

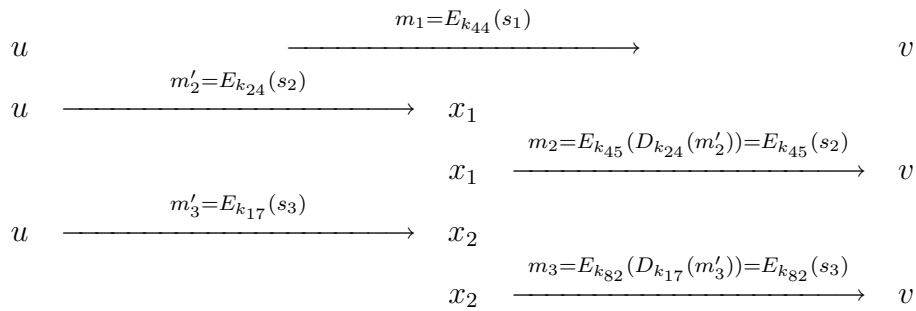
2. Let $I_1 := I_{ux} \cap I'_u = \{17\}$, $I_2 := I_{xv} \cap I'_v = \{82\}$, $z := \min(|I_1|, |I_2|) = 1$

3. Delete IDs from I'_u and I'_v : $I'_u := \{63\}$, $I'_v := \{90\}$

4. $n := n + \text{sgn}(z) = 3$, $z_2 := z_2 + z = 2$, $P := P \setminus \{x\} = \{\dots\}$ (more loops possible)

- Finishes after first loop as $p^w(1, 2) \approx 0.5\% < 1\% = p_0^w$ (and $p_3^w < 0.02\%$).

Assuming that the secret is $S = 11001001$ and u chooses $s_1 = 10010001$, $s_2 = 11110011$ (as well as computes the resulting $s_3 = S \oplus s_1 \oplus s_2 = 10101011$), the following messages are sent:⁹



Finally, v decodes

- $s_1 = D_{k_{44}}(m_1) = D_{k_{44}}(E_{k_{44}}(10010001)) = 10010001$
- $s_2 = D_{k_{45}}(m_2) = D_{k_{45}}(E_{k_{45}}(11110011)) = 11110011$
- $s_3 = D_{k_{82}}(m_3) = D_{k_{82}}(E_{k_{82}}(10101011)) = 10101011$

and reconstructs the secret as $S = s_1 \oplus s_2 \oplus s_3 = 10010001 \oplus 11110011 \oplus 10101011 = 11001001$.

5 Complexity and Performance

As the purpose of the AKE protocol is to achieve secure key establishment while keeping computational and communicational complexity (and thus also energy costs) low, it is sensible to take a look at how well it is actually doing. Note that the establishment of a secret key is treated as an event that does not need to be repeated, as the storage of a few established keys should be within the participating devices' abilities. If, however, this turned out to be a wrong assumption, keys might be made to expire after not having been used for some time, or simply be removed from memory once the space they take up is needed for others.

5.1 Computation

By far the most important calculations involved are the quite inexpensive encryption and decryption of the shares by the sending and receiving nodes as well as the proxies. The required number of such procedures per share is either two for class C_1 , or four for classes C_2 and C_3 , as the respective proxy has to do two additional computations.

⁹ E_k and D_k are meant to be reasonably secure symmetric encryption functions using key k .

5.2 Communication

Giving an expression for the number of network hops traversed by the n secret shares is more difficult, as topological distances between nodes vary. Taking d as the distance in hops between u and v , Zhu et al. determine that altogether, $d \cdot n + 2n_3$ hops are traversed, where n_3 is the number of shares transmitted via paths of class C_3 . Apparently, they assume a protocol that lets the sender transmit the shares intended for the receiver's neighbouring proxies via that receiver itself.

5.3 Trade-Offs

Obviously, the number of shares required is the AKE protocol's deciding factor in all aspects of complexity. Its authors describe a couple of simulative runs of the algorithm given in section 3.3 to determine how this number changes according to different conditions concerning network topology, key pool parameters and the demanded level of security.

All experiments consider a network of 200 nodes of fixed positions, randomly distributed over a $2km \times 2km$ area. Setting a transmission range of $250m$, each node's average number of neighbours is approximately 8.7.¹⁰ In the following, the experiments are summarized by listing the fixed parameters, independent and dependent variables as well as the main result.¹¹ The results of the experiments in this chapter have 95%- confidence intervals that encompass values within 5% of the given ones.

5.3.1 Experiment I

This experiment serves to determine the peer distance d 's influence on the number of secret shares used in each of the three classes of logical paths.

Fixed: $w = 10, p_0^w = 10^{-6}, l = 10000, m = 200$

Indep.: $d = [1..16]$

Dep.: $n \approx 3.8$ and its class-wise composition

As shown in section 4, security depends on the choice of m and l , yet is independent of d , so n is constant throughout the experiment. With the above-mentioned number of 8.7 neighbours per node, C_3 alone could provide all the paths for the 3.8 shares needed on average. One of those, however, is already provided by C_1 , as the probability that two nodes share at least one common key is $1 - (1 - \frac{m}{l})^m$, in this case ≈ 0.98 and generally, as can be seen in figure 2, close to 1 for sensible choices for m and l . As more C_2 paths become available when the distance between peers grows, reliance on C_3 paths concerning the missing 2.8 shares is reduced in favour of C_2 : For $d = 7$ and above, about one share is sent via C_1 and three via C_2 .

¹⁰Zhu et al. found out that beyond a certain value, node density has a negligible impact on n , even less so the absolute size of the network.

¹¹The figures in this section are also taken from Zhu et al.'s paper [1].

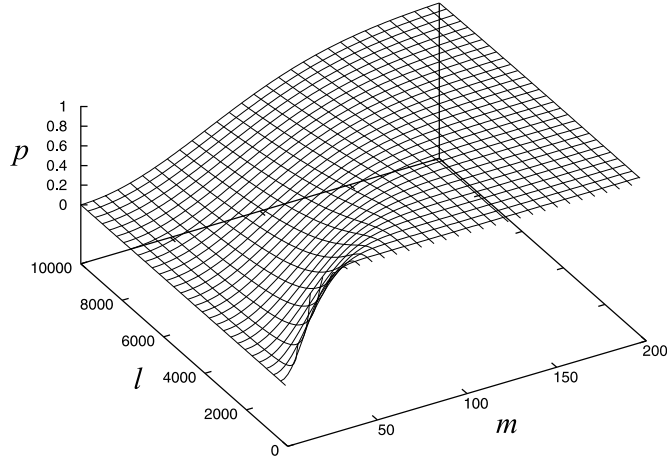


Figure 2: The probability that two nodes share at least one common key

5.3.2 Experiment II

In contrast to the above experiment, this one leaves the peer distance unchanged but varies the size l of the overall key pool as well as the number m of keys assigned to each node by the key server.

Fixed: $w = 10, p_0^w = 10^{-6}$
 Indep.: $l = \{2000, 4000, 8000, 10000\}, m = [100..200]$
 Dep.: n

The number of shares decreases for an increasing m , because a larger choice of keys per node means that more of them can be used in the C_1 share, leading to a better p^w value that, given p_0^w , allows a reduction in C_2 and C_3 shares. This becomes clear when comparing the formulae for p_1^w and p_2^w in section 4: For $z_1 = z_2$, $p_1^w(z_1) < p_2^w(z_2)$.

The effect of l is somehow erratic and apparently depends on the current value of m . Presumably, the m considered are too high to let the total size of the key pool matter.

5.3.3 Experiment III

Again, the number n of required shares is being analyzed, this time over variations of the desired security level p_0^w .

Fixed: $w = 10, l = 10000, m = 200$
 Indep.: $p_0^w = [10^{-8}..10^{-5}]$
 Dep.: n

The number of shares required increases logarithmically with a higher desired security level, i.e. decreasing p_0^w . This is to be expected as, assuming a roughly constant ratio of common

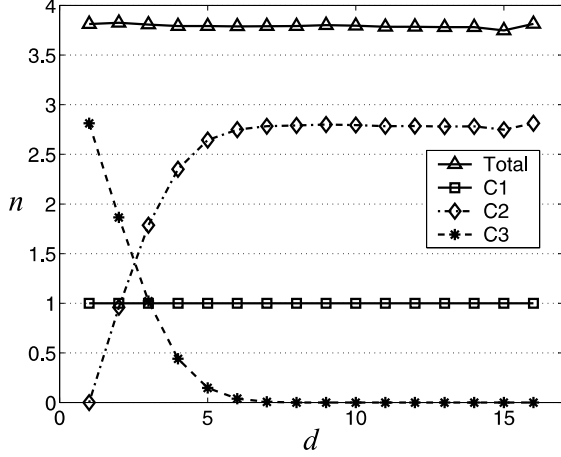


Figure 3: (Exp. I) The number of secret shares' composition as a function of peer distance

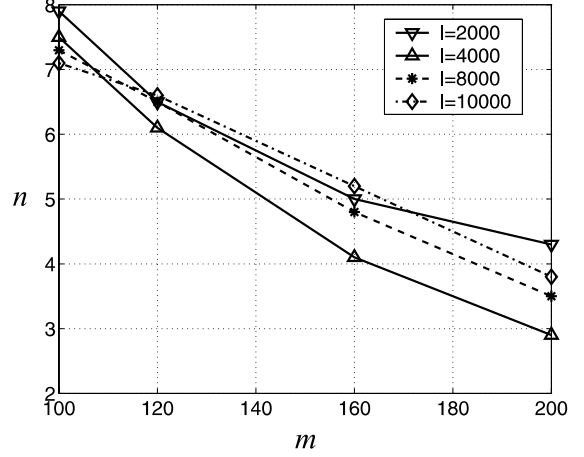


Figure 4: (Exp. II) The number of secret shares required for various parameter combinations

keys per node pair, the numbers of shares and keys are close to proportional,¹² and the latter enters the probability formulae as an exponent.

5.3.4 Experiment IV

The final experiment always assumes the number n of shares to be set to the maximal possible value and graphs the size w of the coalition required in order to discover the secret key, depending on different conditions concerning the key pool size l and the number m of keys per node.

Fixed: $p_0^w = 10^{-6}$

Indep.: $l = \{6000, 10000\}$, $m = \{100, 200\}$, $d = [1..16]$

Dep.: w , (not measured:) $n = n_{max}$

The size of a successful adversarial coalition is required to increase with d : As a small distance severely limits the choice of proxies, that increase is faster initially than it is for higher d , where a certain saturation effect can be observed, i.e. the choice of proxies is no longer the main limiting factor to security.

The impact of l and m varies with peer distance. It is hard to derive clear statements from the given data, as contrary to intuition, higher values for the two variables can apparently lead to worse security at great distances. For the case in which m grows for a constant l , one can understand this behaviour by imagining the extreme case that all nodes share all keys, shrinking the necessary 'coalition' to obtain a secret key to a single node. A comparably large key pool, in turn, reduces the chance that two communicating nodes find the number of common keys

¹²This is not entirely correct, because the algorithm determining the number of shares will have less and less keys to assign to pairs as an increasing portion of them has been used for previous proxies.

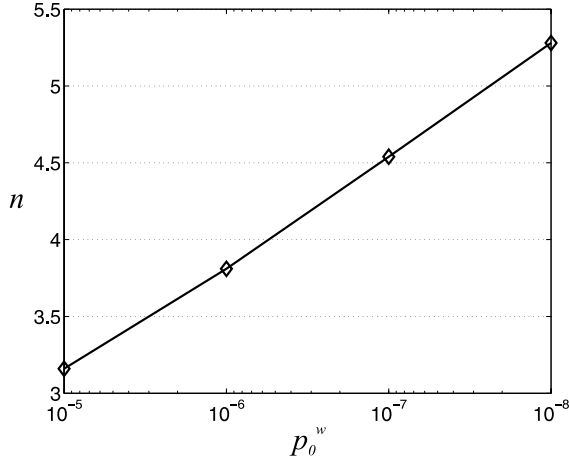


Figure 5: (Exp. III): The number of shares as a function of the desired security level

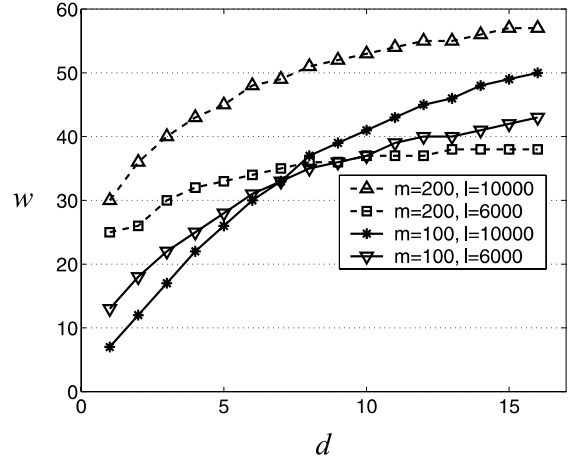


Figure 6: (Exp. IV) The number of colluding nodes necessary as a function of peer distance

needed in order not to be vulnerable against coalitions of certain sizes. Apparently, these effects are minimal for small distances and the given ranges for l and m .

6 Failure and Corruption of Nodes

Obviously, the proposed protocol cannot work when either the sending or receiving node do not behave as expected. There are, however, ways to provide for misbehaving proxy nodes.

6.1 C_2 Proxy Corruption

A C_2 node's failure means that the path chosen by the routing protocol (as the shortest one) is broken and consequently, no shares at all arrive at their destination. Unavailability or malicious behaviour on behalf of intermediate nodes is a general problem for ad-hoc networks and must be addressed by circumventing such nodes following identification of their (in)activity. Whether such attempts were successful can be determined in the AKE protocol's verification step that decodes the `Hello` message: The receiver re-calculates the message's authentication code according to the obtained secret key (just as the sender did) and can thus confidently interpret a match between the received and newly calculated authentication codes as reasonable proof that the key has not been tampered with.

An alternative is the use of several distinct paths to the destination in the first place, using an appropriate routing protocol: These paths could then be used separately (each running its instance of the AKE protocol) or in a combined manner (using a threshold scheme to distribute shares over paths).

In a setting as simple as this, it can directly be seen that v can now determine the linear function $f(x)$ and thus calculate the secret as $S = f(0) = 201$. Note that it is essential for the receiver to know the logical paths' numbers, i.e. pairs $(x, f(x))$ instead of just the function values. These numbers thus have to be appended to the messages m_i .

In case a C_3 proxy alters shares passing through it instead of dropping them, there are two ways to counter this attack: One is to use a one-way *hash* function and to enclose the secret key's hash value with each share to enable the recipient to detect errors before the **Hello** message is transmitted. First of all, as long as the altering proxy does not know the secret key it implicitly produced by changing its share, it cannot send along a fitting hash value – or compute any function on the key, for that matter. Additionally, such a function is called *one-way* because the mapped value (i.e. the secret key) cannot be reconstructed from it; with a proper hash function all possible keys should even remain equally probable despite knowledge of the hash value.

Example 6.2 *Be $S = 1111$ the secret and $h(x)$ the hash function. With $p = 16433$ and $f(x) = [9831x^2 + 12888x + 1111]_p$, three values of f are needed for interpolation. Assuming that $f(2) = 479$ and $f(4) = 12763$ arrive properly, yet $f(6) = 5097$ is changed to 4999, the resulting polynomial $f'(x) = [13927x^2 + 4745x + 1013]_p$ yields a wrong secret $S' = f'(0) = 1013$. If a hash value $h(S)$ was sent along with each share, the error is detected once the receiver computes $h(S')$ because, most likely, $h(1013) \neq h(1111)$. Consequently, a different share like $f(5)$ should be used for correct interpolation.*

The problem remains that in order to find out which proxy delivered a faulty share, many combinations of subsets of the received shares might have to be tried, especially if more than one proxy node modifies its share. If instead, each share were accompanied by its own hash value, checks could efficiently be done on an individual basis.

7 Summary

Ad-hoc networks are dynamically developing, decentralized communication networks of – usually small and portable – electronic devices. Based on a publication by Zhu et al., this paper describes the Ad-hoc Establishment Protocol, a method that allows setting up secret pair-wise keys in such an environment, despite the limitations imposed by conditions like the absence of a central server during operation or the participating nodes' limited communicational and computational capacities.

Naive approaches like public-key cryptosystems (which involve too complex computations) and pre-loading all nodes with a distinct pair-wise key for all others (which leads to too large storage requirements) are dismissed in favour of the concept of probabilistic key sharing: Here, a kind of base keys are distributed over the nodes, admitting some nodes to get the same keys as others. Before a secret mutual communication can take place, the initiating party needs to set

up a secret key and divide it into shares which are then, encrypted with base keys, sent via multiple network paths to the destination node.

The basic idea is that the more base keys are used to encrypt the shares for transmission, the smaller a chance a malign coalition of a certain size stands to be able to reconstruct the secret key. In order not to have this very number limited to the base keys that the two nodes in question have in common, most shares are sent via intermediate nodes that decode and re-encode what they receive.

The key's secrecy is guaranteed only for the case that not even a single of the network's nodes tries to decode it. Given a certain maximum size of malign coalitions and a demand on the level of security (i.e. the probability that the possible coalitions could not determine the secret key), it is possible to determine the number of shares (and thus paths) required and thereby balance security requirements with communication complexity.

One principal extension to the protocol is not to disjunctively divide the secret key into shares but to produce redundant shares in the framework of a threshold scheme, thereby making the method less prone to failure in the presence of failing or sabotaging nodes. Another means of further improving security is to counter the threat of share alterations on the part of intermediate nodes by enclosing hash values.

Whereas the protocol's provision against malign coalitions of a certain limited size is mathematically sound, the practicality of the proposed key server remains doubtful, as its required presence whenever a node joins prohibits an independent and dynamic formation of ad-hoc networks.

References

- [1] S. Zhu, S. Xu, S. Setia, S. Jajodia: *Establishing Pair-wise Keys for Secure Communication in Ad-Hoc Networks: A Probabilistic Approach*, 11th IEEE International Conference on Network Protocols, Atlanta 2003
www.isse.gmu.edu/techrep/2003/03_01.pdf
- [2] C. Mitchell, F. Piper: *Key Storage In Secure Networks*, Discrete Applied Mathematics 21/1988, pp. 215-228
- [3] M. Dyer, T. Fenner, A. Frieze, A. Thomason: *On Key Storage In Secure Networks*, Journal of Cryptology 8/1995, pp. 189-200
www.math.cmu.edu/~af1p/keys.ps
- [4] A. Shamir: *How To Share A Secret*, Communications of the ACM 11/1979, pp. 612-613
www.cs.tau.ac.il/~bchor/Shamir.html
- [5] D. Johnson, D. Maltz, Y. Hu: *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, IETF Internet Draft, 2003
www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt