Secret Instantiation in Ad-Hoc Networks

Sandeep S. Kulkarni
Department of Computer
Science and Engineering
Michigan State University
East Lansing MI 48824

Mohamed G. Gouda
Department of Computer Sciences
The University of Texas
at Austin
Austin, TX 78712

Anish Arora
Department of Computer
and Information Science
Ohio State University
Columbus OH 43210

Abstract

In this paper, we focus our attention on the problem of assigning initial secrets to users in ad-hoc network (respectively, sensors in a sensor network) so that they can use those secrets to ensure authentication and privacy during their communication. The goal of this assignment is to ensure that any two users can communicate securely with each other even though each user maintains only a small number of secrets. With this motivation, we present a protocol that maintains $O(\sqrt{n})$ secrets per user where n is the number of users in the system. We show that our secret distribution protocol suffices for privacy and authentication as well as secure multihop communication between two users. Furthermore, we show that the number of secrets maintained in this protocol is within a constant factor of the optimal. For the case where user capability prevents them from maintaining the necessary secrets, we propose two probabilistic protocols that maintain $O(log\ n)$ secrets and where the probability of security compromise between two users is inversely proportional to the number of secrets they maintain. Thus, our protocols provide a continuum where the level of privacy and authentication depends upon user requirements and capabilities.

Keywords : Ad-hoc networks, Sensor Networks, Security, Instantiating security, Scalability

[1] Email: sandeep@cse.msu.edu, gouda@cs.utexas.edu, anish@cis.ohio-state.edu

Web: `http://www.cse.msu.edu/~sandeep`, `http://www.cs.utexas.edu/~gouda`, `http://www.cis.ohio-state.edu/~anish`

Tel: +1-517-355-2387, +1-512-471-9532, +1-614-292-1836

A preliminary version of this paper appears in Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks, 2004

This work was partially sponsored by NSF CAREER CCR-0092724, DARPA Grant OSURS01-C-1901, ONR Grant N00014-01-1-0744, NSF grant EIA-0130724, and a grant from Michigan State University.

1 Introduction

Security is typically an important issue in wireless ad-hoc networks, including sensor networks, where the communication medium is broadcast in nature and, hence, an adversary can overhear all messages sent by any user. For this reason, a sender must authenticate the receiver and encrypt any messages it sends. One way to achieve this authentication and encryption is to ensure that the sender and the receiver share a common secret that no other user in the network knows.

Another important issue in these networks is multihop communication. Specifically, if two users (respectively, sensors) that need to communicate are not *close* to each other, they may require other users (respectively, sensors) in the network to relay their messages. During such a relay, a user may require that the intermediate users relaying the message cannot learn the contents of the message and that the intermediate users cannot generate messages that incorrectly appear to be from the sender. Hence, we focus on the problem of distributing initial secrets to users so that they can (even on a multihop path) authenticate each other and communicate securely.

In systems with a trusted server, the problem of distributing initial secrets is often handled by assigning each user a secret (e.g., password) that is known only to the user and the trusted server. In these systems (e.g., [1–5]), when two users interact, they use this trusted server to authenticate each other and to establish a secret that is known only to those two users. In many systems, especially in ad-hoc networks, however, this approach is undesirable (respectively, impossible) as no trusted server is available *when two users need to communicate with each other*. Also, in these systems, using certificates signed by a trusted authority is undesirable, as the cost of encryption/decryption using them is often exorbitant. (As an illustration, on MICA motes developed by UC Berkeley, encryption using public key is 100-1000 times slower than symmetric key encryption.) Thus, in these networks, the users must maintain sufficient secrets so that any two users that need to communicate can use these secrets for authentication and/or privacy.

Based on the role of intermediate users in a multihop communication, the solutions for initial key distribution can be classified into two categories. In the first category of solutions (e.g., [6–9]), intermediate users are trusted. Hence, if two non-neighboring users need to communicate, they route the messages through the intermediate users that decrypt and re-encrypt the message. Thus, it suffices that the communicating users share a path such that every user on the path shares a secret with its predecessor and its successor. However, in this case, compromise of a small number of users that act as intermediate users can compromise the security completely.

Another category of solutions includes solutions where intermediate users are not trusted. Hence, if two non-neighboring users communicate the solution guarantees that the intermediate users cannot violate authentication and privacy. In other words, the intermediate users are only responsible for routing the message. However, they should not be able to decrypt it. Clearly, in such a solution, compromise of intermediate users does not affect system security as long as the communicating users share a secret that is not known to the intermediate users. We follow the second approach and our solutions assume that the intermediate users are only responsible for routing the messages, and they are not trusted for decrypting and re-encrypting messages in transit. Thus, given any pair of users, it should be possible to identify a secret that is known only to those two users.

However, the users often have limited memory and limited computing ability. Therefore, the number of secrets they maintain and the amount of computation they perform should be small. Moreover, these networks need to address the problem of scale as the number of users can be high and, hence, the number of initial secrets that each user has should scale with the increase in the number of users.

Based on the above discussion, it follows that one of the important problems for security in ad-hoc networks is to establish a scalable approach for instantiating security so that each user maintains only a small number of secrets even though it can securely communicate (either with certainty or with high probability) with all users in the network.

We present three protocols for instantiating security in wireless ad-hoc networks. In the first protocol, the grid protocol, each user maintains $O(\sqrt{n})$ secrets where n is the number of users. This protocol ensures privacy and authentication between any two users that need to communicate with each other. It also ensures that in a multihop communication, intermediate users cannot learn the contents of the communication they are relaying and that they cannot generate messages that appear to originate from the sender. We also show that the number of secrets maintained in the grid protocol is within a constant factor of the optimal.

Since in large systems, users may not be able to maintain even $O(\sqrt{n})$ secrets, privacy and authentication cannot be guaranteed in them. For such systems, we present two protocols that provide probabilistic security, i.e., in these protocols, the level of security, the probability that the two parties can communicate using secrets that the intruder does not know, is proportional to the number of initial secrets that each user maintains.

Contributions of the paper. The main contributions of the paper are as follows:

- We present a protocol where each user maintains $O(\sqrt{n})$ secrets, where n is the number of users in the system. This protocol guarantees privacy and authentication, i.e., it ensures that the secrets that are used by the communicating parties are not known to any other users in the system. We also show that the number of secrets maintained in our protocol are within a constant factor of the lower bound on the minimum secrets.

- We present two protocols for the case where the level of security is proportional to the number of initial secrets that each user maintains. In these protocols, the probability of a compromise is $a^{(O(m))}$, where $a < 1$ and m is the number of secrets that each user maintains. Thus, in our protocols, a small increase in the number of secrets maintained by a user substantially reduces the probability of privacy compromise. It follows that our probabilistic protocols are especially beneficial for the case where the users do not have the capability to maintain sufficient secrets to ensure privacy. Finally, we show how our probabilistic protocols can resist identity attacks.

Organization of the paper. The rest of the paper is organized as follows: In Section 2, we precisely state the problem of instantiating security in ad-hoc networks. Then, in Section 3, we present our protocol that provides privacy and authentication. In Section 4, we show that the number of secrets maintained in the protocol in Section 3 are within a constant factor of the optimal. In Sections 5.1 and 5.2, we present our probabilistic protocols. In Section 6, we compare these protocols. In Section 7, we identify how our protocols are affected by collusion among users. Finally, we make concluding remarks in Section 8.

2 Model and Problem Statement

As mentioned in the Introduction, each user in the network must know some initial secrets (keys). These secrets will enable the user to obtain privacy and authentication when it communicates with other users. In this section, we identify the precise requirements for the initial secret distribution.

We are interested in protocols where the secrets maintained by a user are independent, i.e., even if an attacker knows a subset of the secrets that a user has, it should not be possible for the attacker to discover the other secrets that user has. In other words, the knowledge of a subset of secrets does not assist the attacker in identifying the remaining secrets through cryptanalytic attacks. Thus, even if two users use a set of secrets to ensure security and the attacker is aware of all but one of those secrets, the attacker cannot compromise that communication.

Furthermore, to compute the space requirement for secrets, we count all secrets that are need to be stored by the user. To illustrate this issue, consider the case where a small number, say x, of secrets are used initially to generate a large number, say y, of *new* secrets by some mathematical manipulation (e.g., using those in evaluating certain polynomials) of the original secrets. In such a case, if these y secrets are stored by the user then the space requirement for this case is y. However, if these secrets are computed on-the-fly then the space requirement is x.

Now, we consider different approaches that may be considered in static networks and argue that these solutions are not suitable for ad-hoc networks and sensor networks. One possible approach for obtaining privacy is to use certificates [10–12] and initially provide each user with a certificate signed by a trusted authority (respectively, a group of trusted authorities). Thus, when two users communicate, they can use these certificates to authenticate each other. However, as discussed in the Introduction, this solution requires high computing power. The drawback of this solution suggests that we should use shared secrets instead of certificates. We consider two simple protocols that use such shared secrets: *single secret protocol* and *full secret protocol*.

Single secret protocol. The simplest protocol that ensures that the sender and the receiver share at least one secret is to establish a single shared secret that all *legitimate* users in the network know. It follows that intruders outside the network cannot learn about the communication between legitimate users. While this approach is currently used in existing sensor networks [13], it is clear that in this protocol, the compromise of one user compromises the security for all. Also, the single secret protocol does not provide any security for multi-hop communication. Specifically, if a user j sends a message to user k via l then j cannot prevent l from learning the contents of that message. Moreover, this protocol cannot be used for providing authentication; in this protocol, the user only knows that it is communicating with *some* legitimate user.

Full secret protocol. Another solution for sharing secrets between a sender and a receiver is to establish a separate shared secret between every pair of users. Although, in this protocol, the compromise of one user does not affect others and intermediate users cannot compromise privacy/authentication while relaying messages, each user needs to maintain n–1 secrets where n is the number of users in the network. In this protocol, if communication between two users is overheard by a third user then the probability of privacy compromise is 0. Thus, when j wants to send a message to k via a user l, l cannot learn the contents of that message. However, this solution becomes infeasible when when the number of users is large and memory associated with each user is low.

Clearly, if we require that the secret shared between two users, say j and k, is not known to any other user in the network, then each user must maintain n−1 secrets where n is the number of users. To reduce the number of secrets, we allow j and k to share a collection of secrets, and require that no other user in the network knows all the secrets in that collection. Clearly, in this situation, it would be possible for j and k to use a combination of these secrets (e.g., by xor-ing these secrets, or by applying some one way hash function to the collection of shared secrets) to ensure privacy and authentication. Thus, we define the problem for guaranteed security as follows:

Problem for *guaranteed* security. Design a secret distribution protocol such that given any two users j and k, they share a collection of secrets such that no other user in the network knows all the secrets in that collection. □

Notation. We use the phrase 'A protocol solves the problem of guaranteed security' if it solves the above problem.

Also, as discussed in the introduction, it may be necessary to provide only probabilistic guarantees about privacy/authentication when the number of keys required to solve the problem of guaranteed security is more than what the users can maintain.

In the context of probabilistic communication, we consider the case where two users, say j and k, are communicating and one other user, say l, is trying to compromise their communication. Hence, we define the problem of probabilistic security as follows:

Problem for *probabilistic* security. Design a secret distribution protocol such that given three randomly selected users, j, k and l, the probability that there is a secret (respectively, a collection of secrets) that both j and k know but l does not know is proportional to the number of initial secrets that j and k have. □

Notation. We use the phrase 'A protocol solves the problem of probabilistic security' if it solves the above problem.

Remark. The problem of probabilistic security could also be posed in the context of two or more attackers working independently (or, working in collusion). While the solutions presented in this paper could be easily evaluated to identify the probability of compromise in such a generalized version, these extensions are outside the scope of the paper.

Intruder/Attacker Model. We assume the standard node-compromise attacker model (e.g., [6,7,9, 14–16]). If a user has been compromised by an attacker then it can utilize all the secrets that the user had. It can do so either passively, i.e., by just listening to messages and attempting to decrypt them if possible. Or, it can do so actively, for example, it can attempt to impersonate another user.

We make no assumptions about mobility in the network. Thus, the users may be mobile or static. We only assume that an orthogonal approach is used to route messages (even in the presence of mobility) and to deal with denial of service attacks. In other words, we only assume that any message sent by legitimate users is delivered (even if users are mobile or the system is subject to a denial of service attack). The approaches used for routing or for dealing with denial of service attacks are outside the scope of this paper.

Initially, we assume that the intruders do not collude. Thus, an intruder is a single non-colluding device that does not share its secrets with any other user. We address the issue of collusions in Section 7. When such collusion is permitted, the colluding users can pool together their secrets in order to break communication security.

3 Guaranteed Security Protocols

In this section, we present the *grid protocol* that extends the solution in [16] and solves the problem of guaranteed security (privacy and authentication). We first present the single grid protocol in Section 3.1. Then, we show how to reduce the number of secrets maintained by a user further while ensuring that the probability of security compromise remains 0. Subsequently, in Section 4, we show that the number of secrets maintained in these protocols are within a constant factor of the optimal.

3.1 Single Grid Protocol

Secret distribution protocol. In this protocol, each user has two types of secrets: *direct secrets* and *grid secrets*: A *direct secret* is shared between exactly two users and each *grid secret* is shared among multiple users. As the name suggests, the *grid secrets* are arranged in a 2-dimensional (2-D) grid.

For example, consider the case where we have 16 users $u_{\langle i,j\rangle}$, $0 < i, j < 4$ (cf. Figure 1). Regarding *grid secrets*, we also maintain 16 secrets $k_{\langle i,j\rangle}$, $0 < i, j < 4$. The users and the *grid secrets* are arranged in a 4x4 grid. Thus, location $\langle i, j\rangle$, $0 < i, j < 4$, is associated with user $u_{\langle i,j\rangle}$ and *grid secret* $k_{\langle i,j\rangle}$. Each user gets the *grid secrets* associated with the nodes in its row and in its column. Thus, in Figure 1, user $u_{\langle 2,3\rangle}$ gets the secrets $k_{\langle 1,3\rangle}$, $k_{\langle 2,3\rangle}$, $k_{\langle 3,3\rangle}$, $k_{\langle 4,3\rangle}$, $k_{\langle 2,1\rangle}$, $k_{\langle 2,2\rangle}$, and $k_{\langle 2,4\rangle}$.

Each user maintains *direct secrets* with users in its row and in its column. Thus, in Figure 1, $u_{\langle 2,3\rangle}$ maintains a separate *direct secret* with $u_{\langle 1,3\rangle}$, $u_{\langle 3,3\rangle}$, $u_{\langle 4,3\rangle}$, $u_{\langle 2,1\rangle}$, $u_{\langle 2,2\rangle}$, and $u_{\langle 2,4\rangle}$. (The *direct secrets* are not shown in the figure.)

Note that a user is aware of its own location in the grid. However, it does not maintain the information about grid locations of other users. Whenever, it needs to communicate with another user, it reveals its grid location (in plain text) and also obtains the grid location of its communicating partner. Based on these grid locations, they determine the secret that should be used to ensure that privacy and authentication are maintained. While an attacker may lie about its location in the grid, as we show in Theorem 3.1, it cannot impersonate another user.

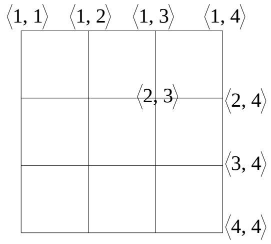


Figure 1: Single grid protocol: A node marked $\langle j, k\rangle$ is associated with user $u_{\langle j,k\rangle}$ and *grid secret* $k_{\langle j,k\rangle}$

Secret selection protocol. When user A wants to communicate with user B and A is not aware of the grid location of B, A sends a request to B that contains its own grid location. Let this location be $\langle j_1, k_1\rangle$. Upon reception of this request message, B also communicates its own grid location to A. Note that this communication is done in plain text and, hence, an attacker listening to this communication can learn its contents. However, this is permissible because our protocol does not assume that the grid locations are secret, i.e., it is provides authentication and privacy even if the attacker knows the grid location of all users.

Now, consider the case where A wants to send message m to B after it has learnt the grid location of B. Let the locations of A and B be $\langle j_1, k_1\rangle$ and $\langle j_2, k_2\rangle$ respectively. In this case, A encrypts m using the following secret selection protocol and sends it along with its own grid location (in plain text).

```
If ($j_1 \neq j_2 \ \wedge \ k_1 \neq k_2$)                  // Users are neither in same row nor in same column
    Use the grid secrets $k_{\langle j_1,k_2\rangle}$ and $k_{\langle j_2,k_1\rangle}$
Else                                       // Users are in the same row or column
    Use the direct secret between $u_{\langle j_1,k_1\rangle}$ and $u_{\langle j_2,k_2\rangle}$
```

As discussed in Section 2, if multiple secrets are selected by the communicating users then a combination of those secrets (using primitives such as xor or by applying a hash function such as MD5). Also, note that the above protocol only focuses on what secrets the users should use. It does not address replay attacks. In other words, an attacker may be able to replay messages that were sent earlier. Several approaches, e.g., nonces and timestamps, have been designed for dealing with replay attacks [4,17]. Any of these approaches could be used in this context.

Theorem 3.1 The single grid protocol solves the problem of guaranteed security.

Proof. Consider the case where two users, say $u_{\langle j_1,k_1\rangle}$ and $u_{\langle j_2,k_2\rangle}$

If $u_{\langle j_1,k_1\rangle}$ and $u_{\langle j_2,k_2\rangle}$ are in the same row (respectively, same column), they use the *direct secret* between them. By definition, no other user has this secret. If $u_{\langle j_1,k_1\rangle}$ and $u_{\langle j_2,k_2\rangle}$ are not in the same row or same column, they use the *grid secrets* $k_{\langle j_1,k_2\rangle}$ and $k_{\langle j_2,k_1\rangle}$. No other user in the network knows *both* these secrets.

Now, if the user $u_{\langle j_1,k_1\rangle}$ uses the above selected secret(s), to communicate with $u_{\langle j_2,k_2\rangle}$, no other user can decrypt that communication. Thus, privacy is guaranteed. Also, when $u_{\langle j_2,k_2\rangle}$ receives this message, it can be sure that no user other than $u_{\langle j_1,k_1\rangle}$ generated that message. Thus, authentication is guaranteed. Note that the properties of privacy and authentication are satisfied even if users are communicating on a multi-hop path. □

Theorem 3.2 In the single grid protocol, each user maintains $2\sqrt{n}-1$ *grid secrets* and $2(\sqrt{n}-1)$ *direct secrets.* □

3.2 Discussion about Grid Protocol

In this section, we discuss some of the questions raised by our probabilistic protocols and briefly discuss some of its extensions. The grid protocol focuses on *what* secrets a user should have instead of *how* it gets those secrets. The way in which a user gets its secrets could depend upon the application at hand. For example, in case of sensor networks, the initial secrets could be provided during the *initial* programming of those sensors, an operation that is typically done in a secure setting. Another approach in this context is to have a trusted authority that is aware of all the secrets. In this approach, a joining user receives its secrets from this trusted authority by a secure channel. One way to achieve such secure channel is to require each user to share a secret with the trusted authority. Alternatively, users could be provided with a public key of the trusted authority. Note that although the use of this public key is expensive (100-1000 times the encryption cost compared with symmetric keys) and undesirable, this design would be acceptable since the public keys are (very) rarely used. In this approach, the trusted authority plays no role when two users need to communicate. Rather, it simply allows a user to obtain (typically in an offline setting) the secrets that it would need for communicating with other users.

Another question in this context is dynamic addition of users in an ad-hoc network. We suggest that a larger grid be used to accommodate a potential increase in the number of users. In such a larger grid, each node would be associated with a secret but only a subset of nodes will be associated with a user. Thus, some users may maintain some secrets that are not useful in communicating with existing users. However, when new users are added to the network, those secrets would be used. In such an approach, a new user would be assigned an unused location in the grid, and it would get the necessary secrets accordingly. Once again, as discussed above, the approach in which a new user obtains these secrets is dependent on the application at hand.

An extension of this protocol would be to allow the grid to grow dynamically. In this case, after a grid is expanded, say by adding a row or column, the existing users would need additional secrets associated with new nodes in its row/column. One approach to provide such secrets is to, intuitively, assign the trusted authority a grid location, say $\langle x, y\rangle$. Thus, although the trusted authority has all secrets used by all users, it behaves as a user located at $\langle x, y\rangle$ when providing new secrets to existing users. Now, the trusted authority can use the secrets as prescribed by the grid protocol to communicate new secrets to existing users. Hence, the trusted authority can guarantee that the new secrets are received only by users that are authorized to have them. Furthermore, as discussed above, application dependent approaches may be used in this context.

As mentioned in Section 2, the grid protocol provides guaranteed security in the absence of colluding users. While we identify the collusion resistance of this protocol in Section 7, we would also like to note that the approach in [16] for maintaining multiple grids in such a way that no two users are in the same row/column in more than one grid could be used to further provide collusion resistance. Finally, the problem considered in this paper, the problem of *instantiating* secrets, is orthogonal to the problem of *maintaining* secrets [18]. In the problem of maintaining secrets, a user changes its shared secrets to thwart an attacker that uses cryptanalytic techniques. While the problem of secret maintenance is outside the scope of the paper, we note that approaches discussed above for distributing the initial secrets could also be tailored to change the *grid secrets* used by different users.

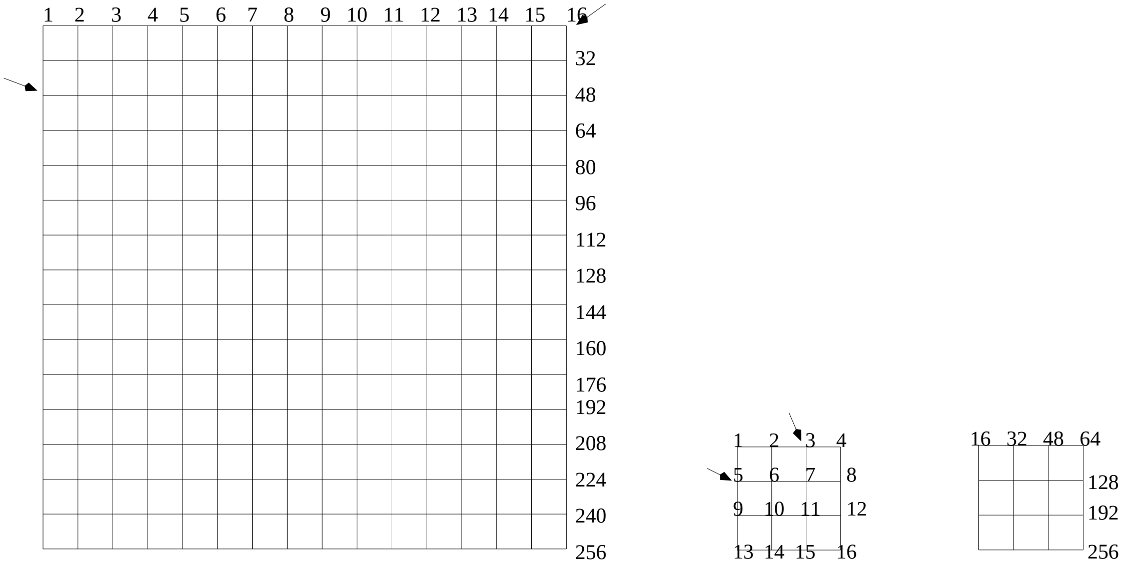
3.3 Hierarchical Grid Protocol

Observe that in the single grid protocol, if two users are in the same row (respectively, column), they maintain a *direct secret*. This *direct secret* is known to exactly two users. In other words, for each row (respectively, column), full secret protocol is used. Thus, we can recursively apply the single grid protocol to reduce the number of such secrets without compromising security. Let m denote the number of users in a row. In the full secret protocol with m users, the number of secrets is $m-1$. By contrast, in the grid protocol with m users, the number of secrets is $4\sqrt{m}-3$. Hence, if $m-1$ is larger than $4\sqrt{m}-3$, it would be possible to reduce the number of secrets further.

To illustrate this approach, consider the 16x16 grid shown in Figure 2 (a). The users in topmost row are numbered as $1, 2, ...16$, and the users in rightmost column are numbered $16, 32, ..., 256$. Based on the single grid protocol, when users 1 and 48 communicate, they use grid secrets at locations marked with an arrow in Figure 2 (a). However, when users 1 and 7 communicate they use the direct secret between them. Instead of maintaining these direct secrets, we rearrange the users in each row and column into a grid. For example, consider Figure 2 (b) and (c), where we have arranged the topmost row and rightmost column into a grid. Once again, we use the grid protocol for these smaller grids and associate *direct secrets* and *grid secrets* with these nodes. Thus, if two nodes that are in the same row in the original grid but are on different row and different column on the subgrid then they can use the *grid secrets* associated with the subgrid. Thus, when users 1 and 7 communicate, they use the grid secrets marked with an arrow in Figure 2 (b). Moreover, when users 1 and 2 communicate, they use the direct secret between them in Figure 2 (b). As we can see, when the number of users in a row is large enough such rearranging will reduce the number of secrets further. Therefore, we can continue this process further until the grid is small enough so that a row cannot be further arranged into a grid. With the above approach, the number of secrets that each user will maintain is approximately equal to numsecrets (we have ignored the -1 factor in this calculation), where

$$\begin{aligned}\text{numsecrets} \quad &= \quad 2\sqrt{n} + 4(\sqrt{\sqrt{n}}) + 8(\sqrt{\sqrt{\sqrt{n}}}) + ... \\ &= \quad (2+\epsilon)\sqrt{n}, \qquad\qquad \text{where } 0 \leq \epsilon \leq 2\end{aligned}$$

Theorem 3.3 The hierarchical grid protocol solves the problem of guaranteed security. □



(a) (b) (c)

Figure 2: Hierarchical Grid Protocol. Arrows in (a) denote the secrets used by 1 and 48, and arrows in (b) denote the secrets used by 1 and 7.

4 Lower Bound for the Problem of Instantiating Security

In this section, we show that the number of secrets maintained by the single grid protocol (respectively, hierarchical grid protocol) is within a constant factor of the optimal. To precisely define this lower bound, we consider the following definitions.

Definition ($secrets_u$): $secrets_u$ denotes the secrets that user u has.

Definition (ms): $ms = max(secrets_u)$

The goal of the lower bound result is to show that in any protocol that solves the problem of guaranteed security, the value of ms in that protocol is at least $\sqrt{n}$. Note that the result allows the possibility that some users maintain a small number of secrets that is considerably less than $\sqrt{n}$. However, some users must maintain at least $\sqrt{n}$ secrets.

To show this result, similar to the definitions of $secrets_u$ and ms, we define $users_k$, mu for each secret as follows.

Definition ($users_k$): $users_k$ denotes the number of users that have secret k.

Definition (mu): $mu = max(users_k)$

Now, we show that if a protocol solves the problem of guaranteed security then $mu \leq ms$.

Theorem 4.1. If a protocol solves the problem of guaranteed security between $n, n > 2$, users then $mu \leq ms$.

Proof. We consider three cases depending upon the value of mu.

- $mu = 1$: In this case, given any two users they have no common secret. Hence, the problem of guaranteed security cannot be solved.

- $mu = 2$: In this case, given any two users they must share a unique secret that is known to only those two users. Hence, the only feasible protocol is the full secret protocol. For the full secret protocol $ms = n - 1$. Moreover, since $n > 2$, the above theorem follows.

- $mu > 2$: Now, consider a secret, say s, that is shared by mu number of users. Let $u_1, u_2, ..u_{mu}$ denote the users that share secret s. Now, we show that user u_1 must maintain at least mu secrets and, hence, $ms \geq mu$.

  To show this, we first observe that $mu \geq 3$. Hence, s cannot be used alone when u_1 and u_2 communicate. In other words, if u_1 and u_2 communicate then they must use at least one additional secret (with/without combining it with s). In other words, to facilitate privacy/authentication between u_1 and u_2, u_1 must maintain at least one extra secret.

  Continuing with this scenario, consider the communication between u_1 and u_3. Based on the above discussion, to facilitate privacy/authentication between u_1 and u_3, u_1 must maintain an additional secret. Moreover, this secret cannot be the same as that shared with u_2; otherwise, u_2 can decrypt communication between u_1 and u_3.

  Based on the above discussion, it follows that for each user $u_2, ..u_{mu}$, u_1 must maintain an additional secret. Combining it with the secret s, we observe that the secrets maintained by u_1 is at least m. Thus, $ms \geq mu$. □

Now, we prove the lower bound on the maximum number of secrets maintained by a user.

Theorem 4.2. If a protocol solves the problem of guaranteed security between $n, n > 2$, users then $ms \geq \sqrt{n}$.

Proof. Note that ms denotes the maximum number of secrets that any user maintains. Now, let j be a user that maintains ms (i.e., the maximum number) secrets. Based on the requirement of instantiating secrets, between every pair of users, there must be at least one shared secret. Based on Theorem 4.1, each secret is maintained by at most ms users. Thus, j can share secret with at most ms^2 users. Since $ms < \sqrt{n}$, j can share a secret with at most $(\sqrt{n}-1)^2$ users. Since $(\sqrt{n}-1)^2 < n$, this is a contradiction. Thus, the maximum number of secrets maintained by a user is at least $\sqrt{n}$. □

Theorem 4.3 The number of secrets maintained in the single grid protocol (respectively, hierarchical grid protocol) is within a constant factor of the optimal. □

Remark. Note that as discussed in Section 2, if two users share only a single secret then they would need $n - 1$ secrets. Hence, to reduce the number of secrets, most users must share at least 2 secrets. Using this observation, it is possible to extend the proof of Theorem 4.2 to show that some users must maintain at least $\sqrt{2n}$ secrets.

5 Protocols for Probabilistic Security

Based on the lower bound identified in Section 4, if users cannot maintain $O(\sqrt{n})$ secrets, then privacy and authentication cannot be guaranteed. To deal with this negative result, we propose solutions for probabilistic security where the probability of privacy compromise is inversely proportional to the number of secrets that users maintain. Before we discuss these protocols, we first introduce the notion of *effectiveness* for a secret distribution protocol.

Definition (Effectiveness). We say that a secret distribution protocol is $\langle s, p \rangle$ effective if the maximum number of secrets that any user has is s and given three randomly selected users, j, k and l, the expected probability that l knows the secret (respectively, all the secrets) used by j and k is at most p.

Definition (Storage dominate). Given two secret distribution protocols, pr_1 and pr_2, with effectiveness $\langle s_{pr_1}, p_{pr_1}\rangle$ and $\langle s_{pr_2}, p_{pr_2}\rangle$ respectively, we say that pr_1 *storage dominates* pr_2 if $s_{pr_1} \leq s_{pr_2}$.

Definition (Security dominate). Given two secret distribution protocols, pr_1 and pr_2, with effectiveness $\langle s_{pr_1}, p_{pr_1}\rangle$ and $\langle s_{pr_2}, p_{pr_2}\rangle$ respectively, we say that pr_1 *security dominates* pr_2 if $p_{pr_1} \leq p_{pr_2}$.

Definition (Dominate). Given two secret distribution protocols, pr_1 and pr_2, with effectiveness $\langle s_{pr_1}, p_{pr_1}\rangle$ and $\langle s_{pr_2}, p_{pr_2}\rangle$ respectively, we say that pr_1 *dominates* pr_2 if $s_{pr_1} \leq s_{pr_2}$ and $p_{pr_1} \leq p_{pr_2}$.

Observation 5.1

- The single secret protocol is $\langle 1, 1\rangle$ effective.
- The full secret protocol is $\langle n-1, 0\rangle$ effective.
- The single secret protocol storage dominates the full secret protocol.
- The full secret protocol security dominates the single secret protocol.
- The single grid protocol is $\langle 4\sqrt{n} - 3, 0\rangle$ effective.
- The single grid protocol (respectively, hierarchical grid protocol) dominates the full secret protocol.

□

Thus, the single secret protocol and the full secret protocol (respectively, single/hierarchical grid protocol) are two extremes for security distribution protocols. In this section, we focus on identifying protocols that are $\langle x, y\rangle$ effective where $1 < x < O(\sqrt{n})$ and $0 \leq y < 1$. Moreover, we focus on security protocols where the level of security is adaptive, i.e., if the number of secrets that each user gets is increased then the probability of a compromise is reduced. We present two such protocols in Sections 5.1 and 5.2. In both these protocols, we focus on the issue of privacy. We show how these protocols can resist identify attacks in Section 5.4.

5.1 Tree Protocol

In this section, we present the first probabilistic protocol, the *tree protocol*, for instantiating security. First, in Section 5.1.1, we present the version of the tree protocol where only one tree is used. Then, in Section 5.1.2, we present the version where multiple trees are used.

For each of these versions, we first identify the secret distribution protocol that determines the secrets that each user should get. Then, we present the secret selection protocol; when two users need to communicate, they use this protocol to determine a shared secret that they should use. Subsequently, we compute the probability of compromise.

5.1.1 Single Tree Protocol

Secret distribution protocol. We organize the secrets in a tree (cf. Figure 3). Each non-leaf node is associated with a secret and each leaf is associated with a user. Each user is assigned an ID that identifies its location in the tree. Each user is provided the secrets along the path towards the root. Thus, user u_1 has the secrets, k_1, k_2 and k_4.

Secret selection protocol. When two users, say, j and k, want to communicate, similar to the grid protocol, they first exchange their identities. Subsequently, they identify their least common ancestor. Based on the secret distribution protocol, the shared secret associated with this ancestor will be available

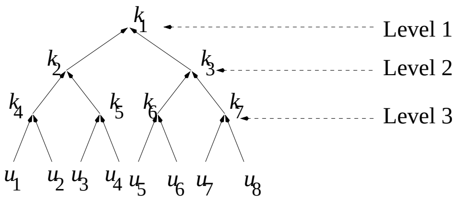


Figure 3: Tree Protocol

to both j and k. Hence, the secret associated with the ancestor will be used for communication between j and k.

For example, if users u_1 and u_2 want to communicate then they will use k_4 whereas if users u_1 and u_5 want to communicate then they will use k_1.

Computing the probability of security compromise. Let l be an intruder that can observe the communication between j and k. We identify the probability that l is aware of the secret that j and k use. During this analysis, let the degree of the secret-tree be d.

Now, we consider different cases based on the shared secret that j and k use during communication. First, we consider the probability that j and k use the secret at the root (level 1). Such a situation arises if k is not a descendant of the level-2-ancestor of j. Thus, the probability of this case is $\frac{d-1}{d}$. And, in this case, the probability that l is aware of the secret that j and k use is 1; all users in the secret-tree have the secret associated with the root.

Next, we consider the probability that j and k use the secret at level 2 in the tree. Such a situation arises if k is a descendant of the level-2-ancestor of j *and* k is not a descendant of the level-3-ancestor of j. Thus, the probability of this case is $\frac{1}{d} * \frac{d-1}{d}$. Moreover, l is aware of the shared secret between j and k iff l is a descendant of the level-2-ancestor of j. Thus, the probability of this case is $\frac{1}{d}$.

Continuing thus, the probability, $p_{compromise}$, that l is aware of the shared secret used by j and k is:

$$\begin{aligned}
p_{compromise} &= \tfrac{d-1}{d}.1(\Sigma_{j=0}^{h}(\tfrac{1}{d})^{2j}) \\
&< \tfrac{d-1}{d}.1(\Sigma_{j=0}^{\infty}(\tfrac{1}{d})^{2j}) \\
&= \tfrac{d-1}{d}\tfrac{1}{1-\frac{1}{d^2}} \\
&= \tfrac{d}{d+1}
\end{aligned}$$

Theorem 5.2 The single tree protocol is $\langle log_d(n), \frac{d}{d+1}\rangle$ effective. □

From Theorem 5.2, it follows that as the degree, d, increases, the level of security decreases. Moreover, as d increases, the number of secrets maintained by a user decreases. Thus, the tree protocol provides the tradeoff between number of secrets maintained by users and the level of security provided to them.

5.1.2 Multiple Tree Protocol

In the single tree protocol, the probability of security compromise is minimized when $d{=}2$. With $d{=}2$, when j and k want to communicate with each other, there is a $\frac{2}{3}$ probability that a third user, l, knows the secret used by j and k. We can reduce this probability further by using multiple secret-trees. We discuss the secret distribution and secret selection protocol with such multiple trees, next.

Secret distribution protocol. In this protocol, the secrets are arranged in multiple trees. Similar to the single tree protocol, in this protocol, each internal node in each tree is associated with a secret and each leaf is associated with a user. Each tree includes all users. For each tree, the user gets the secrets associated with its ancestors in that tree.

Secret selection protocol. For each secret-tree, j and k identify the secret associated with their least common ancestors. Then, they use the combination of all these secrets (e.g., by xor-ing these secrets or by passing those secrets through a one way hash function [19]) during communication. It follows that l can learn the communication between j and k iff l knows all these secrets.

Clearly, if we use two trees where the position of all users is identical and if l knows the secret (used by j and k) in the first tree then, by definition, l will know the secret in the second tree. Hence, when we use multiple trees to reduce the probability of compromise the probability that l knows the secret in one secret-tree should be independent of the probability that l knows the secret in another tree. This can be achieved if there is no correlation between the location of a user across two trees.

Given K secret-trees, each with degree d, the probability that l knows secrets from all the trees is $(\frac{d}{d+1})^K$. Thus, we have

Theorem 5.3 The multiple tree protocol with K trees is $\langle Klog_d(n), (\frac{d}{d+1})^K) \rangle$ effective. □

From Theorem 5.3, it follows that the number of secrets maintained by the tree protocol is O(log n). Moreover, as the number of secrets maintained increases, so does the level of security. Thus, the tree protocol provides the tradeoff between number of secrets maintained by users and the level of security provided to them. In Section 6, we compare this protocol to the grid protocol.

Remark. Note that the above result is applicable for the case where $K << n$. If the number of trees is close to n then the lack of correlation is not possible. Moreover, if the number of trees is close to the number of users then the users would need to maintain a large number of secrets.

As an example, consider the case where we have 2^{10} users and 10 secret-trees of degree 2 are maintained. In each tree, a user maintains 10 secrets and the total number of secrets that a user maintains is 100. In this case, the probability of a compromise is $(\frac{2}{3})^{10} = 1.73\%$. By contrast, if we maintain a separate secret between every pair users then each user will need to maintain $2^{10}-1$ secrets. Multiple tree protocol is even more effective when the number of users is high. For example, if there are 2^{20} users and we maintain 50 secret-trees then the probability of a compromise is $1.56 * 10^{-9}$ when users maintain only 1000 secrets. By contrast, a user would need over a million secrets if we were to maintain a separate secret between every pair of users.

Users with different capabilities. If the users in the network have different capabilities then for that case we can extend the multiple tree protocol. Towards this end, we proceed as follows: Based on the maximum capability of any user, we identify the number of trees used. A user with lower capability will only maintain keys from the first few trees. Now, if j and k communicate and j has lower capabilities than k then they will use only secrets from those trees for which j has maintained the secrets. Thus, when two users with different capabilities communicate, they can obtain a level of security that is proportional to the minimum of their capabilities.

5.2 Complementary Tree Protocol

In this section, we present the second probabilistic protocol, the *complementary tree protocol*. Similar to the tree protocol in Section 5.1, this protocol arranges secrets in a tree (respectively, multiple trees). However, the secret distribution protocol and secret selection protocol are different. Once again, as in Section 5.1, we first present the single complementary tree protocol and multiple complementary tree protocol.

5.2.1 Single Complementary Tree Protocol

Secret distribution protocol. Similar to the tree protocol, we organize the secrets in the tree of degree d. In this protocol, we require that $d \geq 3$ (cf. Figure 4). All nodes in the tree except the root are associated with a secret. Each leaf of the tree is associated with a user. (Note that a leaf is associated with a user as well as a secret.)

The secret distribution is as follows: For each level (except level 1), the user gets secrets associated with the *siblings of its ancestors (including itself)*. Thus, user u_1 gets secrets k_2, k_3 (level 2), k_5, k_6 (level 3), k_{14} and k_{15} (level 4). A user does not get the secrets associated with its ancestors.

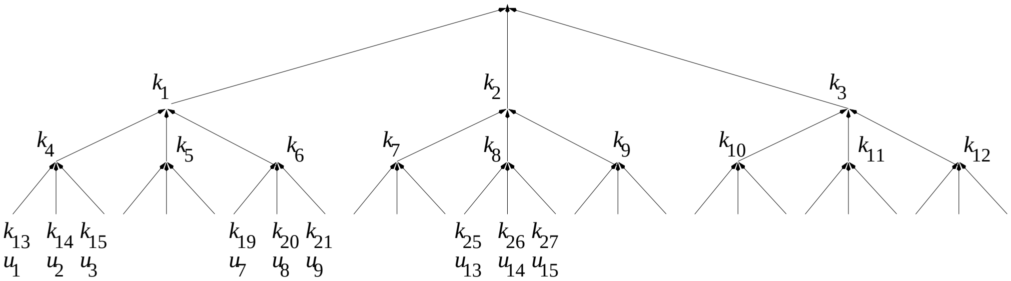


Figure 4: Complementary Tree Protocol

Secret selection protocol. When two users, say j and k, want to communicate, they first identify their least common ancestor. Let z be the least common ancestor of j and k. Let x denote the child of z that is an ancestor of j. Likewise, let y denote the child of z that is an ancestor of k. Now, to communicate, j and k use the secrets associated with all children of z except x and y.

For example, if u_1 and u_2 want to communicate, they use the secret k_{15}. If users u_1 and u_9 want to communicate then they will use the secret k_5. And, if u_1 and u_{15} want to communicate then they will use the secret k_3.

Computing the probability of security compromise. Let l be the intruder that can observe the communication between j and k. We, now, identify the probability that l is aware of the secret(s) used by j and k. Now, consider different cases based on the shared secrets that j and k use during communication. Since no secrets are associated with the root, first consider the case where j and k use the secret(s) at level 2. Such a situation occurs if k is not a descendant of the level-2-ancestor of j. Thus, the probability of this case is $\frac{d-1}{d}$. And, in this case, the probability that l is aware of all the secrets is $\frac{2}{d}$; l knows *all* the secrets used by j and k iff l is a descendant of the level-2-ancestor of j or l is a descendant of the level-2-ancestor of k.

Next, we consider the probability that j and k use the secret at level 3 in the tree. Such a situation arises if k is a descendent of the level-2-ancestor of j and k is not a descendent of the level-3-ancestor of j. Thus, the probability of this case is $\frac{1}{d} * \frac{d-1}{d}$. Moreover, l is aware of the shared secret(s) between j and k iff l is a descendant of the level-3-ancestor of j or l is a descendant of the level-3-ancestor of k. Thus, the probability of this case is $\frac{2}{d} * \frac{1}{d}$.

Continuing thus, the probability, $p_{compromise}$, that l is aware of the secret(s) used by j and k is

$$p_{compromise} \quad = \quad \frac{d-1}{d} \cdot \frac{2}{d} (\Sigma_{j=0}^{h} (\frac{1}{d})^{2j})$$

$$\begin{aligned} &< \frac{d-1}{d}\cdot\frac{2}{d}(\Sigma_{j=0}^{\infty}(\frac{1}{d})^{2j}) \\ &= \frac{d-1}{d}\frac{2}{d}\frac{1}{1-\frac{1}{d^2}} \\ &= \frac{2}{d+1} \end{aligned}$$

Theorem 5.4 The single complementary tree protocol is $\langle (d-1)log_d(n), \frac{2}{d+1}\rangle$ effective. □

5.2.2 Multiple Complementary Tree Protocol

It is possible to reduce the probability of compromise in the complementary tree protocol even further if we maintain multiple trees. More specifically, as in Section 5.1.2, if we maintain K trees where there is no correlation between user locations in different trees, the probability of security compromise will be $((\frac{2}{d+1})^K)$. Thus, we have

Theorem 5.5. The multiple complementary tree protocol with K trees is $\langle K(d-1)log_d(n), (\frac{2}{d+1})^K)\rangle$ effective. □

Similar to the tree protocol, from Theorem 5.5, it follows that the number of secrets maintained by the complementary tree protocol is O(log n). Moreover, as the number of secrets maintained increases, so does the level of security. Thus, the complementary tree protocol also provides the tradeoff between number of secrets maintained by users and the level of security provided to them.

From the above discussion, it follows that the higher the value of d, the lower the probability of compromise. However, when two nodes need to communicate, they need to use a combination of $d-2$ secrets per tree. This suggests that the value of d should be small. In fact, it is desirable to let d be 3 and use multiple trees. To see this, observe that if we maintain one tree with $d=4$, then the probability of security compromise is $\frac{2}{5}$. By contrast, using two trees of degree 3, the probability of security compromise is $(\frac{2}{4})^2$ $(= \frac{1}{4})$. In both cases, the users need to use two secrets during communication. Thus, for the multiple complementary tree protocol, the optimal value of d is 3. (We would like to note that we will reach the same conclusion even if we consider the number of secrets that a user maintains instead of the number of secrets it uses during communication.)

5.3 Discussion about Probabilistic Protocols

In this section, we discuss some of the questions raised by our probabilistic protocols and briefly discuss some of their extensions. Once again, similar to grid protocol in Section 3, the tree protocol also focuses on what secrets need to be distributed rather than *how* those secrets are distributed. Approaches discussed in Section 3, providing the initial secrets during initial programming of sensor networks or providing the initial secrets through the trusted authority, can be used to achieve the secret distribution.

We note that the distribution of keys in the above probabilistic protocols is based on the logical key hierarchy [20] and complementary key hierarchy [21]. The protocols in [20, 21] focus on the problem of maintaining security in a group communication. Hence, in these protocols, the group controller (trusted authority) changes the secrets when the group membership changes so that a leaving (respectively, joining) user cannot access the future (respectively, past) communication.

While the problem considered in this paper is different than the problem considered in [20,21], the results from [20, 21] can be used in solve the problem of secret maintenance or secret revocation (e.g., [22–24]). For example, if some user is compromised and, hence, needs to be removed so that it cannot communicate with other users. In such a case, the trusted authority could use the protocol from [20, 21] to change the secrets that the compromised user shared with others. Moreover, in this case, based on the results in [20, 21], the number of encryptions/messages for performing such secret maintenance for a tree is $O(logn)$.

5.4 Resisting Identity Attacks

So far, in the probabilistic protocols, we have focused on the issue of privacy during communication. Privacy deals with the case where two users need to communicate in such a way that no other user can decrypt the communication between them. For this reason, while designing a solution for privacy, one need not worry about the case where a user *lies about its identity.* In this section, we focus on authentication which needs to deal with the case where a user lies about its identity. As discussed in Section 3, the grid protocol already enables authentication. Hence, we focus on the probabilistic protocols discussed above and the two protocols (single secret protocol and full secret protocol) discussed in Section 2. For each of these protocols, we focus on the issue of authentication, where one user can determine if its communication partner is truly the one it claims to be. For this section, consider the case where user j wants to validate the identity of the user who claims to be k.

We begin with the single secret protocol considered in Section 2. Since this protocol uses only one secret that is shared between all legitimate users, j has no way to authenticate k. The single secret protocol only allows j to conclude that the user it is communicating with is a legitimate user; j cannot verify whether it is k or some other user.

The full secret protocol in Section 2 solves the problem of authentication. If some user l pretends to be k then it will not be able to produce the secret shared between j and k. Hence, when j verifies that its communication partner knows the secret shared between j and k, it can conclude that its communication partner is k.

In the context of the tree protocol (respectively, complementary tree protocol), the user identity is determined by its location in the tree (respectively, all trees). Hence, solving the problem of authentication in the tree protocol is more complex. Specifically, an attacker, say l, can pretend that it is at such a location that it must use the secret associated with the root while communicating with j. Hence, to apply the tree protocol for authentication, we need to ensure that the location of l in one tree is related to its location in another tree. Specifically, if there exists a function f such that given a location x_k of k in one tree, the location of x_k in the next tree is $f(x_k)$ then even if l lies about its location in one tree, it is constrained on the locations that it can use in the next tree. By choosing any function f that permutes the given set of numbers in such a way that correlation between the level of the common ancestor between x_j and x_k and the level of the common ancestor between $f(x_j)$ and $f(x_k)$ is small, we can use the tree protocol for (probabilistic) authentication. Likewise, the complementary tree protocol can be used to provide probabilistic authentication. While identifying such a function f for arbitrary networks remains an open problem, we have found some simple functions that achieve the above property for a small set of users. The problem of identifying such functions is outside the scope of this paper.

6 Comparison of Proposed Protocols

In this section, we compare the protocols presented in Sections 3 and 5. We first analytically compare the tree protocol and the complementary tree protocol. Then, we compute the probability of security compromise in these protocols for different values of n, the number of users in the network. Subsequently, we compare these two protocols with the grid protocol. These results show that a reasonable level of security can be obtained by maintaining a small percentage of secrets maintained by the (single) grid protocol.

Since the number of secrets in the grid protocol are within a constant factor of the minimum number of secrets that users need to maintain, it follows that the tree protocol and the complementary tree protocol are especially useful to provide a reasonable level of security by maintaining a small number of secrets. Finally, we compare the tree protocol and complementary tree protocol with the full secret protocol.

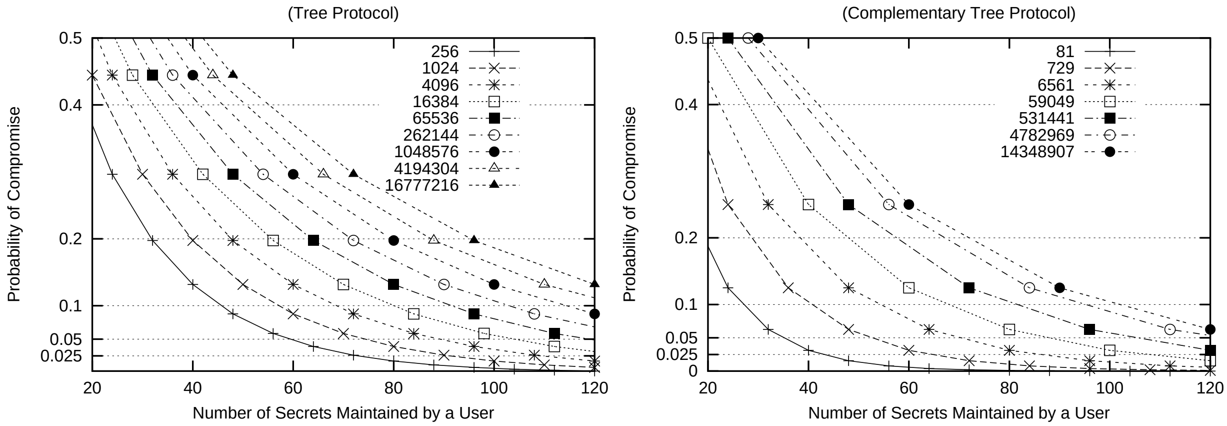
Analytical modeling of probabilistic protocols. Suppose that a user maintains m secrets in the tree protocol. Thus, $\frac{m}{log_2 n}$ trees are maintained (For simplicity, we ignore the issue of partial trees). The level of security with m secrets is $(\frac{2}{3})^{(\frac{m}{log_2 n})}$

Recalling that the degree of the tree in the complementary tree protocol is 3, each user maintains 2 keys per tree per level, and that the probability of compromise by using one tree is $\frac{1}{2}$, if each user maintains m secrets then the probability of compromise is $(\frac{1}{2})^{(\frac{m}{2log_3 n})}$

From the above analysis, in the tree protocol and the complementary tree protocol, the probability of compromise is of the form $a^{O(m)}$, $a < 1$, where m is the number of secrets that a user maintains. In terms of the probability of compromise, the complementary tree protocol is slightly better. In terms of simplicity, however, the tree protocol is better than the complementary tree protocol.

Comparing the probability of compromise in the probabilistic protocols. Now, we compare the probability of compromise for the tree protocol and the complementary tree protocol as we vary the number of users and the number secrets that they maintain. During this comparison, we use the optimal version of these protocols, i.e., we let the degree of the tree to be 2 in the tree protocol and let the degree of the tree to be 3 in the complementary tree protocol.

Figure 5(a) shows the effect of the number of secrets on the probability of security compromise in the tree protocol. As we can see, even if the number of users in the network is large, small number of secrets suffice to ensure that the probability of compromise is small. For example, if we maintain 100 secrets with each user in a network consisting of 64k users, the probability of compromise is less than 10%. Moreover, maintaining additional 20 secrets reduces the probability to approximately 5%.



(a) (b)

Figure 5: Probability of security compromise vs. number of secrets maintained by a user. (Each plot corresponds to the total users in the system, e.g., '256' corresponds to a system where there are a total of 256 users.)

Figure 5(b) shows the effect of the number of secrets on the probability of security compromise in the complementary tree protocol. As we can see, even in this protocol, the probability of security compromise decreases quickly with a small increase in the number of secrets that users maintains.

Figure 6 compares the tree protocol with the complementary tree protocol. As shown in this figure, for the same number of secrets that a user maintains, the probability of compromise in the complementary tree protocol is less than that in the tree protocol.

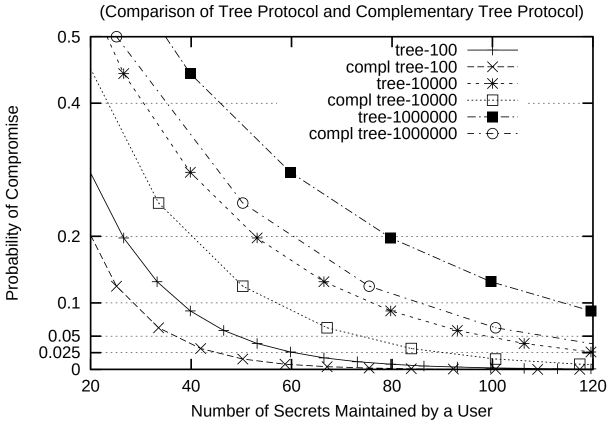
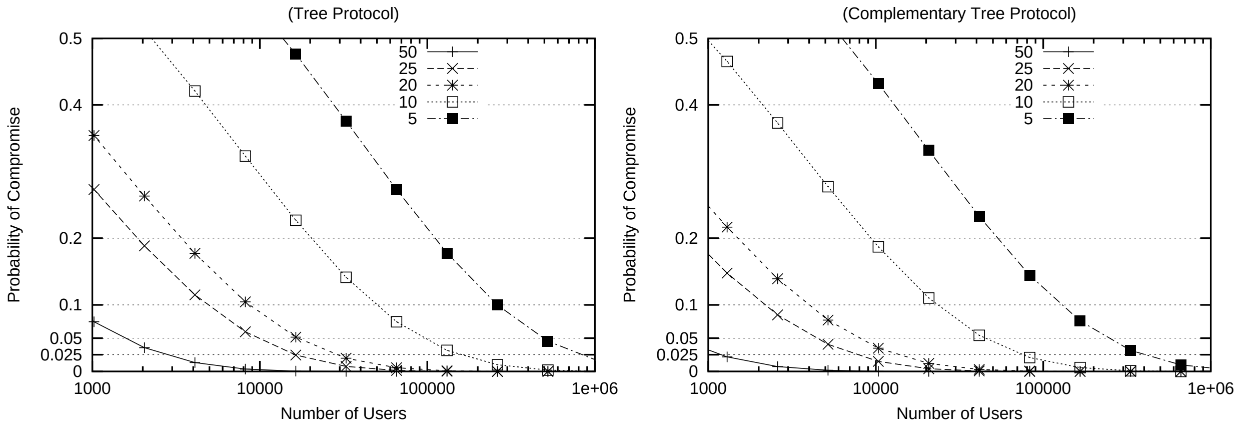


Figure 6: Comparison of Tree Protocol and Complementary Tree Protocol, (Each plot corresponds to the total users in the system, e.g., ‘tree-100’ (respectively, ‘compl tree-100’) corresponds to a system where tree protocol (respectively, complementary tree protocol) is used for a system of 100 users.)

Tradeoff between guaranteed security and probabilistic security. Figure 7 compares the cost of deterministic security versus probabilistic security. More specifically, we ask the question: *How much security could be obtained by using the tree protocol (respectively, complementary tree protocol) if we maintain only a certain percentage of secrets maintained by the grid protocol?* Figure 7 (a) compares the tree protocol and the grid protocol. Based on this graph, we observe that for a group of 10,000 users, maintaining only 20% of the secrets is sufficient to ensure that the probability of security compromise is approximately 6%. Figure 7(b) compares the complementary tree protocol and the grid protocol. From this graph, we observe that for a group of 10,000 users, maintaining only 20% of the secrets is sufficient to ensure that the probability of security compromise is approximately 3%. When the number of users is large, the percentage of secrets required for the same level of security is further reduced. For example, in the complementary tree protocol, for 100,000 users, maintaining only 10% of secrets is sufficient to ensure that the probability of compromise is less than 2.5%.

Comparing the tree protocol and the complementary tree protocol with the full secret protocol. The ability of the tree protocol and the complementary tree protocol to reduce the probability of compromise becomes even more clear if we compare them to the full secret protocol. Figure 8 compares these probabilistic protocols with the full secret protocol. As these graphs show, maintaining only a small percent of secrets is sufficient to keep the level of security compromise low. (Note that in these graphs, we began with 10% of the secrets maintained by the full secret protocol. Then, we reduced this percentage to 0.1%.)



(a) (b)

Figure 7: Probability of compromise if we maintain a certain percentage of secrets used by the grid protocol. (Each plot corresponds to the percentage of secrets maintained, e.g., '50' corresponds to the tree protocol (respectively, complementary tree protocol) that maintains 50% of the secrets maintained by the grid protocol.)

7 Effect of Collusion

In our protocols for instantiating security, so far, we assumed that collusion does not occur, i.e., two or more users do not collaborate (by sharing keys). We compute the effect of the collusion on our grid protocol. Similar analysis can also be used to compute probability of security compromise in the probabilistic protocols in the presence of collusion. We first consider the case where two users collude and then consider a more general case where a collection of w users collude where $w \leq \sqrt{n}$.

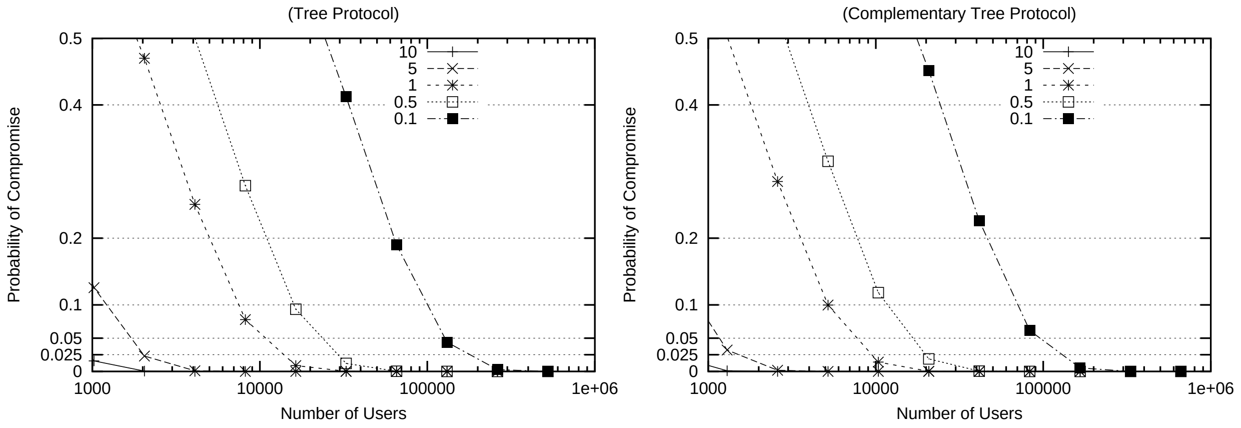
Effect of collusion between 2 users. Consider an example where two users collude in the single grid protocol. Note that a *direct secret* is shared between exactly two users. Hence, if a user leaks that secret to a third user, only the users that shared that *direct secret* are affected. Hence, we only focus on the issue where colluding users share the *grid secrets*. Without loss of generality, we can assume that these users are located in grid locations $\langle 1,1\rangle$ and $\langle 2,2\rangle$ (cf. Figure 1).

For each user j, there are $(\sqrt{n}-1)^2$ users with whom j shares a grid secret. Thus, the number of pairs $\langle j,k\rangle$ that use a grid secret is $n(\sqrt{n}-1)^2$. Now, we identify the number of these pairs that colluding users can affect.

Clearly, users locations $\langle 1,1\rangle$, $\langle 1,2\rangle$, $\langle 2,1\rangle$, and $\langle 2,2\rangle$ cannot communicate with any user securely; the secrets they use are known to the colluding users. Thus, the number of affected pairs in this case is: $4(\sqrt{n}-1)^2$.

For the remaining users in the first row, their communication is compromised when they communicate with any user in the second row (except where *direct secret* is used). (For example, when user $\langle 1,3\rangle$ communicates with user $\langle 2,4\rangle$ communicate, the secrets they use are known to the colluding users.) Since the same scenario applies to the users in the second row (respectively, first and second column), the number of affected pairs in this case is: $4(\sqrt{n}-1)(\sqrt{n}-2))$.

Finally, for all other users (i.e., except those in the first/second row/column), their communication is affected only when they talk with users at locations $\langle 1,1\rangle$, $\langle 1,2\rangle$, $\langle 2,1\rangle$, and $\langle 2,2\rangle$. Hence, the number of affected pairs in this case is $4(\sqrt{n}-2)^2$. Thus, the total affected pairs are $4(\sqrt{n}-1)^2 + 4(\sqrt{n}-1)(\sqrt{n}-2))$



(a)

(b)

Figure 8: Probability of compromise if we maintain a certain percentage of secrets used by the full secret protocol. (Each plot corresponds to the percentage of secrets maintained, e.g., '10' corresponds to the tree protocol (respectively, complementary tree protocol) that maintains 10% of the secrets maintained by the full secret protocol.)

$+\ 4(\sqrt{n}-2)^2$. Thus, probability that pair is affected is $\frac{4((\sqrt{n}-1)^2+(\sqrt{n}-1)(\sqrt{n}-2))+(\sqrt{n}-2)^2)}{n(\sqrt{n}-1)^2}$.

Effect of collusion among w users. We can also extend this result for the case where w users collude, where $w \leq \sqrt{n}$. Without loss of generality, let these users be $\langle 1,1\rangle, ...\langle w,w\rangle$.

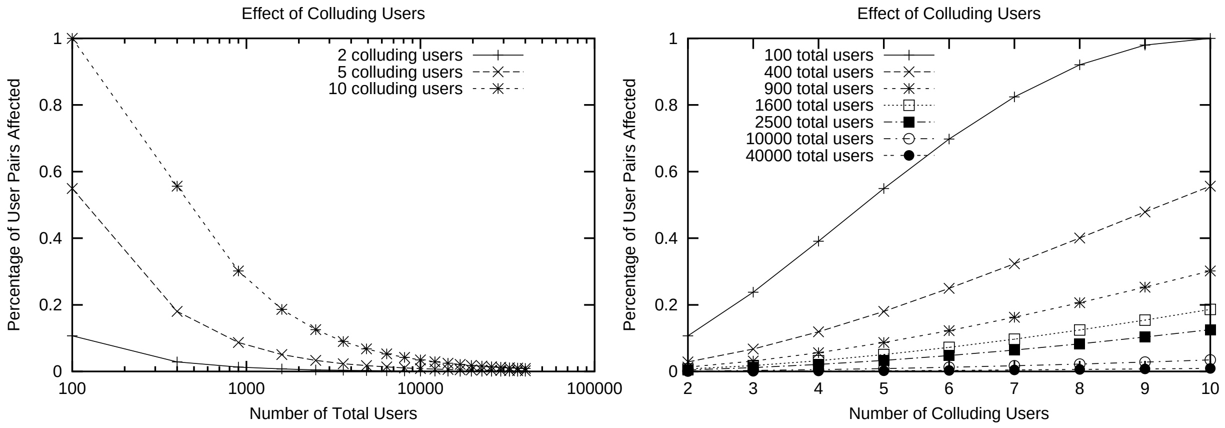
In this case, the users $\langle j,k\rangle$ where $j,k \leq w$ cannot communicate securely with any user. Hence, the number of affected pairs is: $w^2(\sqrt{n}-1)^2$. Likewise, the remaining users in the first k rows will be affected if they communicate with other users in the first k rows (except where *direct secret* is used). Hence, the affected pairs are $2w(w-1)(\sqrt{n}-1)(\sqrt{n}-w)$. Finally, for the remaining users, their communication is affected only when they communicate with users $\langle j,k\rangle$, where $j,k \leq num$. Hence, the number of pairs affected is: $w^2(\sqrt{n}-w)^2$. Combining these numbers, the total number of affected pairs is: $w^2(\sqrt{n}-1)^2 + 2w(w-1)(\sqrt{n}-1)(\sqrt{n}-w) + w^2(\sqrt{n}-w)^2$. In other words, the probability that a pair is affected is $\frac{w^2(\sqrt{n}-1)^2 \ + \ 2w(w-1)(\sqrt{n}-1)(\sqrt{n}-w) \ + \ w^2(\sqrt{n}-w)^2(\sqrt{n}-2)^2)}{n(\sqrt{n}-1)^2}$.

Figure 9 shows the effect of the number of colluding users. As we can see, when the number of colluding users is small, the probability of compromise is low.

8 Conclusion

In this paper, we presented three protocols for instantiating security in ad-hoc networks where each user begins with a set of *initial secrets*. These protocols allow users to obtain privacy and authentication while communicating with each other. Moreover, when two users communicate over a multi-hop path, they can ensure that intermediate users can neither learn the contents of the transmitted messages nor generate messages that incorrectly appear to originate from the sender.

First, we presented the *grid protocol* that ensured that solved the problem of guaranteed security, i.e., it ensured that when two users, say j and k communicate, the set of secrets they use is not known to any other user. This protocol maintained $O(\sqrt{n})$ secrets where n is the number of users in the network. We also showed that the number of secrets maintained by the grid protocol is within a constant factor of optimal (cf. Section 4).



(a)

(b)

Figure 9: Effect of Colluding Users on Grid Protocol

Based on the optimality of the number of secrets maintained by the grid protocol, when the number of users is large and users cannot maintain all these secrets, their only choice is to use to provide probabilistic authentication and privacy. For these cases we presented two probabilistic protocols where the probability of a security compromise is proportional to the secrets they can maintain. In these two probabilistic protocols, the number of secrets maintained by a user is $O(log\ n)$, where n is the number of users. Moreover, as shown in Section 6, in these probabilistic protocols, maintaining a small number of secrets ensured that the probability of security compromise is low.

Our probabilistic protocols can also be tailored to deal with the case where the users have different capabilities. Thus, it is possible that each user maintains secrets that depend on its capability. When two users communicate, their level of security will be decided by the user with lower capabilities.

The results in this paper also provide a tradeoffs encountered in ad hoc networks. Specifically, for the case where the network requires guaranteed security, the grid protocol may be used. For the case where processing power or memory may prevent users from maintaining the secrets required by the grid protocol, they can use the tree protocol (respectively, complementary tree protocol). Moreover, if a subset of users requires a higher level of security than others, they can choose to maintain secrets associated with a larger number of trees. Thus, when these users communicate among themselves, the probability of security compromise will be low. Finally, the tree protocol (respectively, complementary tree protocol) can be combined with the grid protocol. In such a combination, we can partition the users in subgroups. Each subgroup can implement the grid protocol. And, these grids would be used as leaves in the tree protocol. With such an approach, it would be possible to ensure that communication within a subgroup is always secure where the communication across subgroups is secure with some probability.

To improve the security further and to reduce the window of vulnerability, users should use their initial secrets to establish a new disposable secret and use that new secret during further communication. With such a change, security can be compromised only if the adversary can eavesdrop during the establishment of the new secret. Thus, if an adversary moves into the talking range of two users that have established their disposable secret then it would not be able to learn about the communication between them.

There is a small gap between the lower bound on the number of secrets ($\sqrt{2n}$) and the number of secrets maintained ($2\sqrt{n} + o(\sqrt{n})$) in the hierarchical tree protocol. The question of whether this gap can be closed remains open.

References

[1] A. Perrig, R. Szewczyk, J. Tyger, V. Wen, and D. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, 8:521–534, 2002.

[2] M. Tatebayashi, N. Matsuzaki, and D.B. Newman Jr. Key distribution protocol for digital mobile communications systems. *Advances in Cryptology*, 1990.

[3] V. Varadharajan and Y. Mu. Design of secure end-to-end protocols for mobile systems. *Wireless*, 1996.

[4] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of ACM*, 21:993–999, 1978.

[5] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tyger. Spins: Security protocols for sensor networks. *International Conference on Mobile Computing and Networks*, pages 189–199, 2001.

[6] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. *IEEE Symposium on Security and Privacy*, 2003.

[7] L. Eschenauer and V. Gilgor. A key management scheme for distributed sensor networks. *ACM Conference on Computer and Communications Security (CCS)*, pages 41–47, 2002.

[8] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. *ACM Conference on Computer and Communications Security (CCS)*, pages 42–51, 2003.

[9] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. *ACM Conference on Computer and Communications Security (CCS)*, pages 52–61, 2003.

[10] J. Kong, P. Zefros, H. Luo, S. Lu, and L. Zhang. Providing probust and ubiqutious security support for mobile ad-hoc networks. *IEEE International Conference on Network Protocols*, 2001.

[11] J. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad-hoc networks. *ACM Symposium on Mobile Ad Hoc Networking & Computing*, 2001.

[12] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network*, 13(6), 1999.

[13] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: Link layer security for tiny devices. Available at `http://www.cs.berkeley.edu/~nks/tinysec/`, 2003.

[14] R. Blom. Non-public key distribution. *Advances in Cryptology: Crypto*, pages 231–236, 1982.

[15] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly secure key distribution for dynamic conferences. *Advances in Cryptology*, pages 344–355, 1992.

[16] Li Gong and David J. Wheeler. A matrix key-distribution scheme. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 2(1):51–59, 1990.

[17] J. Steiner, C. Neuman, and J. Schiller. Kerbores: An authentication service for open network systems. *Proceeds of Winter Technical Conference USENIX*, pages 191–202, 1988.

[18] V. Naik, S. Bapat, A. Arora, , and M. Gouda. Whisper: Local secret maintenance in sensor networks. *Workshop on Principles of Dependable Systems*, 2003.

[19] R. Rivest. The MD5 message-digest algorithm. *RFC 1321, Internet Activities Board*, 1992.

[20] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 2000.

[21] S. S. Kulkarni and B. Bruhadeshwar. Adaptive rekeying for secure multicast. *IEICE/IEEE Joint Special Issue on Assurance Systems and Networks, Transactions on Communications*, 2003.

[22] Jianying Zhou, Feng Bao, and Robert H. Deng. Validating digital signatures without ttp's time-stamping and certificate revocation. *Information Security, 6th International Conference*, pages 96–110, 2003.

[23] Johannes Blomer and Alexander May. Key revocation with interval cover families. *Selected Areas in Cryptography*, 2001.

[24] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication. *RFC 2104*, 1997.