Cryptographic Protocols, Sensor Network Key Management, and RFID Authentication

by

Jiang Wu

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2009

Abstract

This thesis includes my research on efficient cryptographic protocols, sensor network key management, and radio frequency identification (RFID) authentication protocols.

Key exchange, identification, and public key encryption are among the fundamental protocols studied in cryptography. There are two important requirements for these protocols: efficiency and security. Efficiency is evaluated using the computational overhead to execute a protocol. In modern cryptography, one way to ensure the security of a protocol is by means of provable security. Provable security consists of a security model that specifies the capabilities and the goals of an adversary against the protocol, one or more cryptographic assumptions, and a reduction showing that breaking the protocol within the security model leads to breaking the assumptions. Often, efficiency and provable security are not easy to achieve simultaneously. The design of efficient protocols in a strict security model with a tight reduction is challenging.

Security requirements raised by emerging applications bring up new research challenges in cryptography. One such application is pervasive communication and computation systems, including sensor networks and radio frequency identification (RFID) systems. Specifically, sensor network key management and RFID authentication protocols have drawn much attention in recent years.

In the cryptographic protocol part, we study identification protocols, key exchange protocols, and ElGamal encryption and its variant. A formal security model for challenge-response identification protocols is proposed, and a simple identification protocol is proposed and proved secure in this model. Two authenticated key exchange (AKE) protocols are proposed and proved secure in the extended Canetti-Krawczyk (eCK) model. The proposed AKE protocols achieve tight security reduction and efficient computation. We also study the security of ElGamal encryption and its variant, Damgård's ElGamal encryption (DEG).

Key management is the cornerstone of the security of sensor networks. A commonly recommended key establishment mechanism is based on key predistribution schemes (KPS). Several KPSs have been proposed in the literature. A KPS installs pre-assigned keys to sensor nodes so that two nodes can communicate securely if they share a key. Multi-path key establishment (MPKE) is one component of KPS which enables two nodes without a shared key to establish a key via multiple node-disjoint paths in the network. In this thesis, methods to compute the k-connectivity property of several representative key predistribution schemes are developed. A security model for MPKE and efficient and secure MPKE schemes are proposed.

Scalable, privacy-preserving, and efficient authentication protocols are essential for the success of RFID systems. Two such protocols are proposed in this thesis. One protocol uses finite field polynomial operations to solve the scalability challenge. Its security is based on the hardness of the polynomial reconstruction problem. The other protocol improves a randomized Rabin encryption based RFID authentication protocol. It reduces the hardware cost of an RFID tag by using a residue number system in the computation, and it provides provable security by using secure padding schemes.

Acknowledgements

I am very grateful for the support and guidance given by my supervisor Professor Douglas R. Stinson. I also wish to thank the other members of my examination committee for their numerous comments and helpful suggestions that have improved the quality of this thesis. My doctoral studies have been partly supported by an NSERC Postgraduate Scholarship, a David R. Cheriton Graduate Scholarship in Computer Science, and a President's Graduate Scholarship. I would like to gratefully acknowledge all of this assistance. Finally, I appreciate the love and support from my families.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Key exchange, identification, and public key encryption are among the fundamental protocols studied in cryptography. There are two important requirements for these protocols: efficiency and security. Efficiency is evaluated using the computational overhead to execute a protocol. In modern cryptography, one way to ensure the security of a protocol is by means of provable security. Provable security [76, §1]consists of a security model that specifies the capabilities and the goals of an adversary against the protocol, one or more cryptographic assumptions, and a reduction showing that breaking the protocol within the security model leads to breaking the assumptions. Often, efficiency and provable security are not easy to achieve simultaneously. The design of efficient protocols in a strict security model with a tight reduction is challenging.

Security requirements raised by emerging applications bring up new research challenges in cryptography. One such application is pervasive communication and computation systems, including sensor networks and radio frequency identification (RFID) systems. Specifically, sensor network key management and RFID authentication protocols have drawn much attention in recent years.

In this thesis, the following topics are studied: identification protocols, authenticated key exchange protocols, security proofs for some public key encryption schemes, key management protocols for sensor networks, and RFID authentication protocols.

1.1 Cryptographic Protocols

1.1.1 Computational Assumptions

Let $G = \langle g \rangle$ be a finite cyclic group of order q with generator g, let q be the order of G, and let $\lambda = log_2|G|$ be the *security parameter*. Let $x \in_R S$ indicate choosing x uniformly at random from the set S. Some cryptographic problems and assumptions are listed below:

- *Discrete log (DL) problem*: given $X \in_R G$, find $x \in \mathbb{Z}_q$ such that $g^x = X$. We use DLG$(\cdot)$ to denote the function that solves the DL problem.

- *Computational Diffie-Hellman (CDH) problem*: given $X \in_R G$ and $Y \in_R G$, find g^{xy}, where $g^x = X, g^y = Y$. We use CDH$(\cdot,\cdot)$ to denote the function that solves the CDH problem.

- *Decisional Diffie-Hellman (DDH) problem*: distinguish (g^x, g^y, g^z) from (g^x, g^y, g^{xy}) where $x \in_R \mathbb{Z}_q, y \in_R \mathbb{Z}_q, z \in_R \mathbb{Z}_q$. We use DDH$(\cdot,\cdot,\cdot)$ to denote the function that solves the DDH problem. DDH(X, Y, Z) outputs 1 if $Z =$ CDH(X, Y). Otherwise, DDH(X, Y, Z) outputs 0.

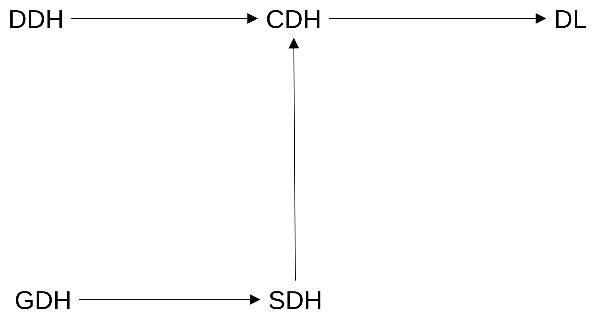
- *Strong Diffie-Hellman (SDH) problem*: given a pair of elements (g^a, g^b) where $a \in_R \mathbb{Z}_q, b \in_R \mathbb{Z}_q$, find the element $C = g^{ab}$ with the help of a *restricted Decision Diffie-Hellman Oracle* $DDH_{g^a}(g^c, g^d)$, which outputs "1" if $g^d = g^{ac}$, or "0" if $g^d \neq g^{ac}$. The restricted Decision Diffie-Hellman Oracle DDH$_{g^a}(\cdot,\cdot)$ can be considered as a DDH oracle DDH$(\cdot,\cdot,\cdot)$ where the first input is fixed to g^a.

- *Gap Diffie-Hellman (GDH) problem*: given a DDH oracle that solves the DDH problem, solve the CDH problem.

Given a function $f(\cdot)$, if for any polynomial $Q(\cdot)$, if holds that $f(\lambda) < 1/Q(\lambda)$ for λ large enough, then we say that $f()$ is *negligible* (in λ).

The *DL/CDH/DDH/SDH/GDH assumption* says that the probability that any polynomial time (in λ) algorithm can solve the DL/CDH/DDH/SDH/GDH problem is negligible (in λ).

The known relations among the assumptions are as follows: the DDH assumption implies the CDH assumption, which in turn implies the DL assumption; and the GDH assumption implies the SDH assumption, which in turn implies the CDH assumption. The relations are illustrated in Figure 1.1.



$$\begin{tikzcd}
\text{DDH} \arrow[r] & \text{CDH} \arrow[r] & \text{DL} \\
\text{GDH} \arrow[r] & \text{SDH} \arrow[u] &
\end{tikzcd}$$

Figure 1.1: Relation between the DL/CDH/DDH/SDH/GDH assumptions.

It is believed that the DDH assumption (therefore CDH and DL assumptions) holds in certain groups. Two such groups are:

1. Prime order subgroups of the group $\mathbb{Z}_p^*$ where p is a large prime.
2. Elliptic curve groups $E/\mathbb{F}_p$ where $\mathbb{F}_p$ is a finite field and the order of the group is prime and the embedding degree of $E(F_q)$ is not small.

A detailed discussion on DDH and related problems can be found in [20].

GDH was proposed in [84] and SDH was proposed in [3]. It is believed that GDH and SDH also hold in the above two prime order groups [84], [3].

1.1.2 Identification

An *identification protocol* enables one entity (a *prover*) to identify itself to another (a *verifier*) as the legitimate owner of some key or some identifying value. In the public key setting, a prover has a *private key* and the corresponding *public key*. The prover needs to prove that it holds the private key, and the verifier uses the public key to verify the proof.

The objective of an adversary against an identification protocol is to impersonate the prover. Based on the communication model, the adversary is allowed various types of attacks. In the smartcard communication model considered in [41], the adversary can launch *sequential attacks*, where the adversary plays the role of verifier and interacts with the prover in multiple identification sessions sequentially. Using the information collected in the process, the adversary then tries to impersonate the prover to other verifiers. Many classic identification protocols, e.g., [43], [51], [83], [92], are designed in this model. In the Internet communication model considered in [10], the adversary is capable of *concurrent attacks*, where the adversary can concurrently interact with many instances of the prover while it is impersonating the prover to a verifier. Concurrent attacks cover sequential attacks. Another powerful attack, a *reset attack*, where the adversary can reset the provers to their previous states, was considered in [7]. A formal security model named CR2 (Concurrent-Reset 2) was proposed in [7] to capture both concurrent attacks and reset attacks. Four paradigms to build identification protocols secure in the CR2 model were also proposed in [7]. In [103], an efficient challenge-response identification protocol is proposed and proved to be secure against *active intruder attacks* and reset attacks, which are considered to be equivalent to attacks in the simplified CR2 model.

In this thesis, we present our work on identification protocols [112]. We simplify the CR2 model in [7] for deterministic challenge-response identification protocols. Then we propose an extremely simple identification protocol and prove its security in the simplified CR2 model.

1.1.3 Authenticated Key Exchange

An *authenticated key exchange* (AKE) protocol enables two parties to establish a shared session key with the property that, when one party computes a session key, it is ensured that only the intended party can compute the same session key. In a typical AKE protocol, each party holds a *static private/public key* pair, which will be used in all key exchange sessions. In each session, each party generates an *ephemeral private/public key* pair for this session. The two parties exchange their public keys and compute the session key using the public keys and their private keys.

Several AKE security models have been proposed to capture attacks against an AKE protocol. Typical early work includes the BR model by Bellare and Rogaway [10], and the CK model by Canetti and Krawzcyk [22]. Recently, LaMacchia, Lauter and Mityagin presented an *extended Canetti-Krawczyk (eCK) security model* [68]. In the eCK model, the adversary controls the communications among the parties, and is able to reveal private keys of some parties and session keys of some sessions. The objective of the adversary is to learn some information of a target session key. There are certain limitations on the attacks so that the adversary does not achieve its goals by using some unreasonably powerful attacks. For example, the adversary is not allowed to reveal the target session key, or to reveal both the static and ephemeral private keys of one party in the target session. The eCK model covers the BR model and CK model. It also covers some attacks not considered in the BR model or the CK model.

Several AKE protocols are proved secure in the eCK model, e.g., the CMQV protocol [106] and the NAXOS protocol [68]. Roughly speaking, their proofs say that, if an adversary can break the protocol with some probability ϵ, then with probability $c\epsilon$ ($c < 1$), some hard problem can be solved, which is assumed infeasible. The c value is one indication of the *tightness* of the proofs. It is desirable that c is close to 1, which means the proof is tight. In the eCK model, the proof of NAXOS is tighter than CMQV.

Efficiency is also important to AKE protocols. Efficiency is evaluated by using the computational overhead of one party in establishing a session key. MQV [69] and its variants HMQV [65] and CMQV [106] are among the most efficient AKE protocols. However, MQV does not have a security proof based on commonly used assumptions. HMQV has a very complicated security proof in the CK model (plus some security properties not captured by the CK model) and it relies on some non-standard assumptions. CMQV has a relatively simple security proof in the eCK model. Both proofs are not as tight as that of NAXOS. However, NAXOS is not as efficient as MQV. It has been an open problem to find an AKE that achieves the performance of MQV and has a security reduction as tight as that of NAXOS.

In this thesis, we present two AKEs named SMEN ($\underline{\text{S}}$ecure $\underline{\text{M}}$QV or $\underline{\text{E}}$fficient $\underline{\text{N}}$AXOS) and SMEN$^-$ [116]. Both protocols are secure in the eCK model and enjoy the same simplicity and tightness in security proof as NAXOS. The efficiency of both protocols is close to that of MQV.

1.1.4 ElGamal Encryption

The ElGamal encryption scheme [38] is one of the classic public key encryption schemes. One description of the scheme is as follows. Let G be a cyclic group of prime order q, and let g be a generator of G. The private key is $a \in \mathbb{Z}_q$ and the public key is $A = g^a$. The ciphertext of a message $m \in G$ is ($c_1 = g^r, c_2 = mA^r$). In decryption, m is computed as $m = c_2/{c_1}^a$.

In an attack against a public key encryption scheme, an adversary is given a target ciphertext and needs to get some information of the corresponding plaintext. Three attack models are often used:

- *Chosen-plaintext attacks (CPA).* The adversary can choose arbitrary plaintexts and obtain the corresponding ciphertexts.
- *Non-adaptive chosen-ciphertext attacks (CCA1).* The adversary can choose arbitrary ciphertexts and obtain the corresponding plaintexts before it is given the target ciphertext.
- *Adaptive chosen-ciphertext attacks (CCA2).* After being given the target ciphertext, the adversary can choose arbitrary ciphertexts different from the target ciphertext and obtain the corresponding plaintexts.

ElGamal encryption is provably secure under CPA, and is insecure under CCA2. It is conjectured to be secure under CCA1, but there has been no formal proof.

In [30], Damgård proposed a variant of ElGamal encryption (DEG). In this scheme, the private key is a pair $(a, b) \in \mathbb{Z}_q \times \mathbb{Z}_q$, and the public key is a pair (A, B) where $A = g^a, B = g^b$. The ciphertext of a plaintext $m \in G$ is (g^r, A^r, mB^r). To decipher (X, Y, Z), the private key owner outputs Z/X^b if $Y = X^a$, or rejects the ciphertext as invalid if $Y \neq X^a$. In [30] and [49], DEG is proved to be CCA1 secure based on certain assumptions.

In this thesis, we present our work in [115], where we investigate the connection between the security of ElGamal encryption and some cryptographic assumptions. We also give a new proof that DEG is CCA1 secure under certain assumptions. The proof is simpler than previous proofs.

1.2 Sensor Network Key Management

Distributed sensor networks (DSNs) consist of large numbers of wireless sensor nodes. In general, a sensor node is battery powered, equipped with sensors, data processing units of limited computation capability, limited memory space, and short-range radio communications. The sensor nodes are usually distributed randomly in a certain area for data acquisition and environment monitoring. After deployment, they operate unattended and without physical protection. They need to communicate with each other to accumulate data and (possibly) relay the data to a base station. In many applications, such as battle field surveillance, communications between sensor nodes have to be encrypted. At the same time, sensor nodes deployed in a hostile environment are prone to be captured and compromised.

Because sensor nodes have limited computation and communication capabilities as well as limited power supply, and they are often deployed in a random way, techniques for cryptographic key establishment in conventional networks are not suitable for DSNs. Public key techniques are computationally expensive and are generally not recommended for use in sensor nodes. Symmetric key distribution schemes such as Kerberos cannot be used due to the fact that we cannot assume that there is an online trusted server that can be accessed by the sensor nodes.

A commonly recommended key distribution approach for DSNs is *key predistribution*, which installs cryptographic keys or key material in sensor nodes before they are deployed. Later, after the sensor nodes are deployed, they establish pairwise keys with their neighbouring nodes (i.e., within the wireless communication range of the nodes). Then communications between nodes are encrypted using the established pairwise keys.

A key establishment scheme consists of three phases, namely key predistribution, shared-key discovery, and path-key establishment [40].

1. *key predistribution*

   Key predistribution takes places before the sensor nodes are deployed. In this phase, keys or key materials are installed into each node.

2. *shared-key discovery*

   The shared-key discovery phase takes place after the sensor nodes are deployed in the operational environment. In this phase, every node discovers its neighbors in wireless communication range with which it shares keys.

   The shared-key discovery phase establishes the topology of the sensor network as seen by the routing layer of the DSN. A link exists between two sensor nodes only if they share a key, and if a link exists between two nodes, then all communication on that link is secured by link encryption and authentication.

3. *path-key establishment*

   In the path-key establishment phase, two nodes within each other's wireless communication that did not discover any shared key in the shared-key discovery phase establish a key via intermediate nodes. For two nodes A and B that do not share a key, they can establish a *path key* using a *secure multi-hop path* between them. On such a path, each two adjacent nodes have a secure link. A can transport a key K to B via this path. On each hop, K is transported using the secure link, encrypted using the key shared by the two adjacent nodes.

1.2.1 Connectivity of Key Predistribution Schemes

There has been extensive research on key predistribution schemes (KPSs) for DSNs. In [40], Eschenauer and Gligor first introduced the idea of securing a DSN with a random KPS, where keys assigned to a sensor node are randomly chosen from a large key pool, and two nodes can establish a pairwise key if they share at least one key. A generalized scheme by Chan, Perrig, and Song [26] stipulates that two nodes can establish a pairwise key if they share at least q common keys. Other schemes based on random assignment of keys include Liu, Ning and Li [73] and Du *et al.* [37], where each node is assigned shares of a random subset of bivariate polynomials (chosen from a pool of polynomials of pre-specified degree), and two nodes can establish a pairwise key if they have shares on a common polynomial.

Instead of randomly assigning keys or key material, Çamtepe and Yener first proposed deterministic methods using combinatorial designs (a.k.a. set systems) in key predistribution [24]. The combinatorial structures they considered were symmetric balanced incomplete block designs and generalized quadrangles. Some other KPSs based on combinatorial designs include Wei and Wu's scheme [108] using difference families and all k-subsets of a set; Lee and Stinson's scheme [70], [71] based on common intersection designs (in particular, transversal designs); and Chakrabarti, Maitra and Roy's scheme [25] which merges blocks in a set system.

There are two graphs associated with a DSN based on a KPS. In both of these graphs, the vertex set is just the nodes in the DSN. The *block graph* of a KPS is the graph where two vertices are adjacent if the corresponding sensor nodes can establish a pairwise key. For the KPSs we consider in this thesis, this is equivalent to saying that the two nodes share at least one common key. The *geometric graph* of the DSN is the graph in which two nodes are adjacent if they are within each other's communication range (after the nodes have been deployed). Two nodes are able to communicate directly if and only if the corresponding vertices are adjacent in both the block graph and the geometric graph. This motivates consideration of the *network graph*, which is the intersection of the block graph and the geometric graph.

In order to study the connectivity of a DSN secured by a KPS, Eschenauer and Gligor used the Erdös-Rényi random graph model [39]. In this model, the probability that the DSN is connected is estimated to be

$$P_c = e^{-e^{\ln n - np}}$$

where n is the number of nodes and p is the probability that two nodes are within each other's communication range and can establish a pairwise key. Several other works, including [26], [37], [108], [59], used the same approach. By applying the random graph model, it is implicitly assumed that the network graph is a random graph, which in turn implies that the block graph and the geometric graph both behave like random graphs.

In [33], Di Pietro *et al.* raised questions about the above-mentioned assumptions, arguing that it is not realistic because the geometric nature of the DSN is not taken into account. They proposed another approach to derive the probability that the DSN is connected. Their approach is based on the assumption that a node's communication radius r is constant and the number of nodes n is infinite. This is different from the more common model for random geometric graphs where r decreases as n increases, which we are more interested in.

There also has been some research on the connectivity of the block graph of a random KPS, without assuming it is a random graph. In [34], Di Pietro *et al.* proved that if $p = c \log n/n$ for $c > 8$ (along with some other conditions), where p is the probability that two nodes in the block graph of a random KPS are adjacent, then the block graph is connected (for n large enough) with high probability. In previous studies, the block graph is often assumed to be a random graph, in which case $c > 1$ is a sufficient condition for the graph to be connected. It would be desirable to improve the bound $c > 8$ to a tighter bound, close to $c > 1$, while relaxing the assumptions.

An object closely related to the block graph of a random KPS is the *random intersection graph*. A random intersection graph $G(n, v, p)$ consists of n nodes. For each node, every point in a pool of size v is independently assigned to the node with probability p. Two nodes are adjacent if they have at least one common point. Research on random intersection graphs can be found in [97], [44], [50], [14].

Given that a graph is connected, κ-connectivity can measure how strong the connectivity is. A graph is *κ-edge-connected* if removal of any $\kappa - 1$ edges does not disconnect the graph. A graph is *κ-vertex-connected* if removal of any $\kappa - 1$ vertices does not disconnect the graph. We note that κ-connectivity has been studied in the random graph model (see [18]) and the random geometric graph model ([88]). Bettstetter [13] studied κ-connectivity of the geometric graph of a DSN, computing the probability $\Pr[\kappa \geq x]$ for given x where κ is given as the minimum node degree d_{min}. Tague and Poovendran [104] studied κ-connectivity of DSNs secured by KPSs under two assumptions: 1.) the sensor nodes are distributed according to a two-dimensional Poisson point process, and 2.) a KPS-secured DSN is a random geometric graph. So far, there has been no study of κ-connectivity for block graphs of KPSs.

In previous research, the performance of a DSN has been evaluated by the connectivity probability, denoted P_c, of the DSN. There are several limitations to this approach. First, this kind of analysis only indicates if a DSN is likely to be connected; it does not determine how strong the connectivity is, which may be more important in practice. As suggested by random graph theory, the transition of a random graph from being disconnected to being connected is a "short phase" [86]. In a practical application, it results that a DSN is more likely to be "strongly connected" than "just barely connected". Therefore, we would like to know how strong the connectivity is. Second, P_c is affected by the way that the vertices are geometrically located, which is independent of the KPS. This suggests that it is of interest to also consider the connectivity of the block graph, which depends only on the KPS underlying the DSN.

Another problem we address is whether the Erdös-Rényi random graph model is suitable for estimating the connectivity probability of the network graph of a DSN. In the network graph, denoted G_N, the existence of an edge is dependent on other edges, while in the random graph model, any edge is independent of all other edges. So it is clear that G_N is not a random graph. However, when using the random graph model to study DSNs (as in [40], for example), it is implicitly assumed that G_N behaves like a random graph. Although many previous papers have used that assumption implicitly, it would be desirable if we did not have to rely on the assumption. If this is not possible, then it would be desirable for clarity to make all required assumptions explicit and validate them computationally through simulations.

As mentioned previously, the network graph is the intersection of the block graph and the geometric graph. We denote these graphs as G_N, G_B and G_G, respectively. In this thesis, we study the κ-edge-connectivity of G_B and G_N. κ-edge-connectivity of G_B is an indication of the suitability of the KPS, and the κ-edge-connectivity of G_N indicates how strongly the resulting DSN is connected. For DSNs based on random KPS, we use simulations to determine if the κ-edge-connectivity of G_B and G_N can be estimated using the random graph model.

1.2.2 Multipath Key Establishment

To enhance the security of a path key, Chan, Perrig and Song [26] and Zhu *et al.* [117] proposed to use multiple paths to transmit key shares. Suppose that there are m secure paths between A and B. A could send n key shares $s_1, \ldots, s_n$ to B, one share via each path. B recovers the key K as $K = s_1 \oplus \cdots \oplus s_n$. Note that the actual number of such paths can be estimated using the k-connectivity properties of a sensor network secured by key predistribution schemes (see, e.g., [111]).

The path key establishment using $K = s_1 \oplus \cdots \oplus s_n$ is vulnerable to message dropping or altering. In [117], Zhu *et al.* also proposed to use an (n, k) secret sharing scheme [93] to compute the shares and to recover the key. An (n, k) secret sharing scheme generates n

shares for a secret s. With any k of these n shares, the secret s can be recovered. The secret sharing scheme enables B to recover the key when some shares are dropped. Secret sharing schemes also provide error correcting ability so that the key can be recovered when some shares are modified (see [75, 90, 100]). However, the error-correcting ability of the (n, k) secret sharing scheme is not used in [117].

To withstand message dropping and altering attacks, Huang and Mehdi [57] proposed a *multi-path key establishment scheme* (the HM scheme) based on Reed-Solomon (RS) codes. In the HM scheme, A chooses a key and encodes it in an RS codeword which consists of multiple symbols. The symbols are sent to B via multiple paths. The RS code provides error-correction ability so that B can recover the key when some symbols are dropped or altered.

Deng and Han [32] proposed another RS code based multi-path key establishment scheme named JERT (Just Enough Redundancy Transmission). JERT is designed for two neighbouring nodes which have a direct but insecure communication link, over which B can send feedback to A. Unlike the HM scheme where A transmits all symbols of a codeword, in JERT, A transmits the symbols incrementally. When B has received enough symbols and recovers a key, B and A can run an authentication protocol over their direct link to verify the recovered key. If the key recovered by B is correct, then A will not send the remaining symbols. Because the communication over the direct link does not involve other nodes as in the code symbol transmission, the transmission overhead of the authentication protocol is considered free.

In this thesis, we first define a model for multi-path key establishment (MPKE). This model enhances the model used in [57] and [32]. We have two specific security objectives:

reliability

The adversary nodes should not be able to prevent B from computing the key K that was chosen by A.

secrecy

From the point of view of the adversary nodes, the entropy of K (given the information that they observe) should be sufficiently high so that they cannot compute K.

The above objectives can be realized using a protocol for *perfectly secure message transmission* (PSMT). Constructions and bounds for PSMT have been studied extensively since the 1993 paper of Dolev *et al.* [36]. PSMT was first suggested for use in multipath key establishment in sensor networks in 2004 by Wang [107].

We propose a new optimal protocol for one-round PSMT based on Reed-Solomon codes. Our protocol is somewhat similar to a protocol found in Fitzi *et al.* [45]. Then we use our

PSMT protocol to obtain two new multi-path key establishment schemes that can be applied provided that fewer than one third of the paths contain an adversary node. Our first scheme works in the same setting as the HM scheme, where A does not need to receive feedback from B. Our second scheme works in the same setting as JERT, where A can receive feedback from B to reduce message transmission. We optimize the parameters of both our proposed schemes so that A uses the minimum transmission possible for B to recover a secure key.

Finally, we describe another MPKE scheme that tolerates a higher fraction (less than 1/2) of paths controlled by the adversary. This scheme is based on a new protocol for a weakened version of message transmission, which is very simple and efficient.

1.3 RFID Authentication

Radio Frequency Identification (RFID) is an automated object identification technology. RFID systems consist of two main components: tags and readers. Tags are small radio transponders. They contain the identification information of objects to which they are attached. Readers query these tags for the identifying information about the objects. Readers often have secure access to a back-end database. For simplicity, a reader and a back-end database can be treated as a single entity.

While being promising in a wide range of applications such as supply chain, libraries, and anti-counterfeiting, RFID also raises privacy and security concerns. Since RFID tags respond to radio interrogation automatically, malicious scanning of tags is a plausible threat. Even if the information emitted by a tag is encrypted, the information may be used to track the tag, thus causing privacy issues. An equally significant problem is authentication. One purpose of RFID tags is to prove the authenticity of objects. If an RFID tag can be scanned and replicated, then a counterfeit tag can be made to impersonate the authentic one.

RFID protocols must provide privacy and security under those possible attacks. A common attack model is as follows. There is an adversary who is able to eavesdrop on the communications between the tags and readers, interrogate the tags, compromise some tags, reset tags, and change the messages between the tags and readers. The goal of the adversary is to impersonate or track uncompromised tags. Correspondingly, an RFID protocol needs to meet two requirements: be secure against impersonation and be untraceable.

In addition to the security and privacy requirements, an RFID protocol needs to be scalable on the reader side and efficient on the tag side. An RFID system may have a large number of tags. The RFID protocol must be scalable to allow the reader to deal with such a large number of tags. On the other hand, a tag has very limited resources in computation and memory, so the protocol must be efficient on the tag side. For cryptographic tools, symmetric key techniques are usually considered feasible for some RFID tags while public key techniques are considered unsuitable for any of them.

To design an RFID protocol satisfying all the desired privacy, security, scalability, and efficiency properties is challenging. Privacy makes RFID authentication different from conventional cryptographic authentication. Using symmetric key techniques, secure authentication relies on a symmetric key shared between a tag and reader. For privacy, a tag cannot identify itself to a reader before an authentication interaction, thus the reader does not know which key to use in the interaction. A straightforward solution is to try every key. This is prohibitively costly when the number of tags becomes large. This is known as the *key search problem*. Literature in this area has sought to reduce the cost of key search. Every such protocol proposed so far involves some kind of tradeoff among the desired properties [61].

Several existing approaches to the key search problem are as follows.

- *Tree approach.* In [77], Molnar and Wagner proposed a scheme to reduce the key search cost to $\Theta(\log n)$ where n is the total number of keys. The scheme uses d sets of keys $K_1, \ldots, K_d$. Each set contains b keys. Each tag is assigned with d keys $k_1 \in K_1, \ldots, k_d \in K_d$. The key assignment can be represented as a tree of depth d and each node has b children. The scheme can accommodate up to b^d tags in total. In an identification session, the tag runs d rounds of interaction with the reader. In the i^{th} round, the tag uses the key k_i, and the reader searches K_i. In a session, the reader needs to search through db keys.

  Since a key will be used in more than one tag, compromise of one tag results in compromise of keys in other tags; hence this leads to privacy infringements [4].

- *Synchronization approach.* The basic idea in the synchronization approach is for the reader and tags to maintain a synchronized state. For example, every tag T_i maintains a counter c_i. On interrogation, the tag outputs $E = f_{k_i}(c_i)$ where f is a keyed hash function and k_i is a secret key shared between the reader and T_i, then increase c_i. The reader can compute all possible outputs of all tags and store the results in a searchable table. In each interrogation, the reader searches the response from the tag in the table.

  There are several variants of the above approach in the literature, e.g., Ohkubo, Suzuki, and Kinoshita [82], Henrici and Müller [56], Juels [60], and Dimitriou [35].

- *Time-space tradeoff approach.* In [5] and [4], Avoine, Dysli, and Oechslin used a time-space trade-off to achieve $\Theta(n^{\frac{2}{3}})$ in both memory and time complexity for key search. Time-space tradeoff is used as an enhancement to the synchronization approach. The basic idea is to organize all future response values of all tags in chains. Each chain consists of a sequence of response values as follows. The reader chooses a random function f to map a response value x to a pair (i, j), then the response of tag i at the j^{th} query will be the successor of x in the chain. Only the head and the tail of a chain is stored. When receiving a response x, the reader can apply f to get its successors. The reader searches x and its successors in all the tails of the chains to locate the chain which contains x. Then it starts from the head of the chain, applies f repeatedly, and locates (i, j) of x.

- *Public key approach.* If public key cryptosystem can be used, then it would be easier to solve the key search problem. Whether a public key cryptosystem can be implemented on RFID tags remains an open problem and has drawn much effort. In [85], Oren and Feldhofer proposed WIPR, an RFID identification protocol based on a randomized Rabin encryption scheme. WIPR is very efficient in hardware, requiring only 5705 gates, and its design aims at strong security and privacy requirements. While Oren and Feldhofer provided an implementation of the protocol, they also note that some details of the protocol design have not been fully analyzed.

In addition to the conventional privacy requirement, two stronger privacy features for RFID systems, termed *backward untraceability* and *forward untraceability*, were also considered in the literature. Backward untraceability means that, if the adversary reveals the internal state of a tag at time τ, the adversary is not able to tell whether a transaction before time τ involves the tag. Forward untraceability means that, if the adversary reveals the internal state of a tag at time τ, the adversary is not be able to tell whether a transaction after time $\tau + \delta$ (for some $\delta > 0$) involves the tag, provided that the adversary does not eavesdrop on the tag continuously after time τ.

Our work in RFID authentication protocols includes three contributions:

1. We propose a novel RFID protocol that provides security, privacy, scalability, and efficiency [113]. The protocol is based on polynomial computation, which has not been used in RFID protocol design before. The protocol achieves the desirable properties at the cost that a tag is allowed to be interrogated no more than a given number of times. This is similar to some previous protocols such as [82], but the interrogation threshold in our protocol is much higher. In previous protocols, the threshold increases linearly with the memory or computation overhead, while in our scheme, the threshold increases *quadratically* with the overhead. Besides, the threshold in our protocol also increases linearly with the communication overhead. Overall, our protocol achieves

a very high interrogation threshold using moderate computation, memory, and communication resources, and within this interrogation threshold, the protocol is secure and untraceable. Furthermore, the computation and memory complexity on the RFID reader/server side is logarithmic in the number of RFID tags.

2. We analyze the security and privacy features of WIPR, a public-key RFID authentication protocol [85], and propose two variants with improved security and hardware efficiency [114]. We show that a reduced version of WIPR is vulnerable to short padding attacks and reset attacks. We discuss countermeasures to avoid these attacks by properly specifying the details of the protocol. Then we propose two variants, WIPR-SAEP and WIPR-RNS, to provide better security and to further reduce the hardware cost. WIPR-SAEP uses an additional hash function to achieve provable security. WIPR-RNS uses a residue number system (RNS) computation to reduce the hardware cost of WIPR. WIPR-RNS may also provide better security guarantees in that it uses standard cryptographic primitives instead of the non-standard ones in WIPR. The two changes (SAEP padding and RNS computing) can be used independently or combined together.

3. We analyze an RFID identification scheme which is designed to provide forward untraceability and backward untraceability. We show that if a standard cryptographic pseudorandom bit generator (PRBG) is used in the scheme, then the scheme may fail to provide forward untraceability and backward untraceability. To achieve the desired untraceability features, the scheme can use a robust PRBG that provides forward security and backward security.

Chapter 2

Cryptographic Protocols

In this chapter, we study identification protocols, authenticated key exchange protocols, and ElGamal encryption and its variant DEG (Damgård's ElGamal encryption). In Section 2.1, we review the CR2 model from [7] and present a simplified CR2 model for challenge-response identification protocols in Section 2.1.1, and we present a new identification protocol and its security proof in Section 2.1.2.

In Section 2.2, we review the eCK model in Section 2.2.1 and efficient algorithms for exponentiation in Section 2.2.2, then present two new protocols SMEN and SMEN$^-$ in Section 2.2.3 and Section 2.2.4 respectively.

In Section 2.3, we present a security analysis for ElGamal encryption in Section 2.3.1 and a security analysis for DEG in Section 2.3.2.

2.1 Identification Protocols

2.1.1 Security Model

Model for General Identification Protocols

In [8], Bellare *et al.* defined a formal security model for identification protocols. The model captures concurrent attacks, where an adversary can concurrently execute identification protocols with multiple provers while he impersonates a prover to a verifier, and reset attacks, where an adversary can reset a prover to its previous state. The model is named CR2 (concurrent-reset 2). A weaker model is named CR1, where an adversary can concurrently execute identification protocols with multiple provers only *before* he impersonates a prover to a verifier.

The CR2 model consists of a description of an identification protocol and a security game that defines the security of the protocol. An identification protocol is described by a function $ID()$ which specifies how all associated processes (key generation, message computation, session id or decision computation) are implemented. The first argument to $ID()$ is a keyword (keygen for key generation, prvmsg for prover message computation, vfmsg for verifier message computation, prvsid for prover session id computation, vfend for verifier end message and session id computation) which invokes the subroutine responsible for that function on the other arguments. The protocol consists of $m(k)$ moves where k is a security parameter and $m()$ is a polynomial. $m(k)$ is odd so that the first and last moves belong to the prover. When it is necessary, a START message is added for the prover. The corresponding security game is defined in Game 1.

Model for two-move Protocol with Deterministic Prover

We make the following changes to adapt the CR2 model for two-move identification protocols with a deterministic prover.

1. change the condition for $\hat{A}$ to win. $\hat{A}$ wins only when $decision = accept$ and $sid_V \notin SID_1 \cup \cdots \cup SID_p$.

   In the CR2 model, $\hat{A}$ wins if $\hat{A}$ makes the verifier accept without simply relaying messages between the verifier and a prover (this condition is captured by condition 1, $decision = accept$ and $sid_V \notin SID_1 \cup \cdots \cup SID_p$), or if $\hat{A}$ makes two prover instances output the same sid (this condition is captured by condition 2, there exist $1 \leq a < b \leq p$ with $SID_a \cap SID_b \neq \emptyset$). We note that condition 2 is not essential for the security of an identification protocol because no verifier is cheated here. In addition, condition

```
Initialization:
1. $(pk, sk) \leftarrow ID(\textsf{keygen}, k)$                    //key generation
2. Choose tape $R_V$ for verifier at random
3. $C_V \leftarrow 0$                                   //message counter for the verifier
4. $p \leftarrow 0$                                   //number of active prover instances

Execute adversary $\hat{A}$ and reply to its oracle queries as follows:
When $\hat{A}$ makes query $\textsf{WakeNewProver}$                //activate a new prover instance
1. $p \leftarrow p + 1$; $SID_p \leftarrow \emptyset$; Pick a tape $R_p$ at random
2. Return $p$
When $\hat{A}$ makes query $\textsf{Send}(\textsf{prvmsg}, i, msg_1 \parallel \cdots \parallel msg_{2j})$ with $0 \leq 2j \leq m(k)$ and $1 \leq i \leq p$
1. $msg_{2j+1} \leftarrow ID(\textsf{prvmsg}, sk, msg_1 \parallel \cdots \parallel msg_{2j}; R_i)$
2. $s \leftarrow msg_{2j+1}$
3. If $2j + 1 = m(k)$ then
    - $sid \leftarrow ID(\textsf{prvsid}, sk, msg_1 \parallel \cdots \parallel msg_{2j+1}; R_i)$,
    - $s \leftarrow s \parallel sid$
    - $SID_i \leftarrow SID_i \cup \{sid\}$
4. Return $s$
When $\hat{A}$ makes query $\textsf{Send}(\textsf{vfmsg}, msg_1 \parallel \cdots \parallel msg_{2j-1})$ with $1 \leq 2j - 1 \leq m(k)$
1. $C_V \leftarrow C_V + 2$
2. If $2j < C_V$ then
    - return $\perp$                                //not allowed to reset the verifier
3. If $2j - 1 < m(k) - 1$ then
    - $msg_{2j} \leftarrow ID(\textsf{vfmsg}, pk, msg_1 \parallel \cdots \parallel msg_{2j-1}; R_V)$
    - return $msg_{2j}$
4. If $2j - 1 = m(k)$ then
    - $sid_V \parallel decision \leftarrow ID(\textsf{vfend}, pk, msg_1 \parallel \cdots \parallel msg_{2j}; R_V)$
    - return $sid_V \parallel decision$

$\hat{A}$ wins the game if either of the following is true:
1. $decision = accept$ and $sid_V \notin SID_1 \cup \cdots \cup SID_p$
2. There exist $1 \leq a < b \leq p$ with $SID_a \cap SID_b \neq \emptyset$
```

Game 1: CR2

2 makes some identification protocols without known flaws insecure in CR2, e.g., the signature based challenge-response protocol in Section 3 of [8]. We consider condition 2 to be too strong a condition and drop this condition to simplify the model.

2. change the description of the protocol from odd-number of moves to two moves.

   This change is straightforward. With this change, for each prover instance i, $\hat{A}$ will only make query $\mathsf{Send}(\mathsf{prvmsg}, i, x; R_i)$ once, with $x = msg_1$; for the verifier, $\hat{A}$ will make query $\mathsf{Send}(\mathsf{vfmsg}, x)$ twice, the first time with $x = \perp$ and the second time with $x = msg_1 \parallel msg_2$. Note that the use of the message counter C_V ensures that $\hat{A}$ must query $\mathsf{Send}(\mathsf{vfmsg}, \perp)$ before querying $\mathsf{Send}(\mathsf{vfmsg}, msg_1 \parallel msg_2)$. The resulting game is described in Game 2.

3. change the game for a deterministic prover.

   Since the prover is deterministic, the random tapes R_i for prover instance i can be removed. Then the sets SID_i can be merged into one set SID, and the prover instance indicator i and related operations can be removed. The resulting game is described in Game 3.

In Game 3, query $\mathsf{Send}(\mathsf{vfmsg}, msg_1)$ must be made after the query $\mathsf{Send}(\mathsf{vfmsg})$. Query $\mathsf{Send}(\mathsf{prvmsg}, i, msg_1)$ can be before or after $\mathsf{Send}(\mathsf{vfmsg})$, and it can be made a polynomial (in k) number of times. $\mathsf{Send}(\mathsf{vfmsg}, msg_1)$ is the last query. Game 3 can be rewritten into an interactive game described in Game 4. In Game 4, n_1 and n_2 are polynomial in the security parameter k. We specify that, in the identification protocol, $sid = msg_1 \parallel msg_2$ and $sid_V = msg_1 \parallel msg_2$ (This specification is only for the security proof and does not have to be included in the protocol itself). Since $msg_2 \leftarrow ID(\mathsf{prvmsg}, sk, msg_1)$, the same msg_1 values result in the same msg_2 values, and hence the same sid values. Therefore, $sid_V \notin SID$ in Game 3 is equivalent to $c \neq c_i$ in Game 4.

We then rewrite Game 4 to Game 5. It can be shown that the difference between the probabilities that $\hat{A}$ can win in Game 4 and Game 5 is negligible. Then, Game 5 is a security definition equivalent to the CR2 game (with a modification on the conditions that $\hat{A}$ wins) for challenge-response identification protocols with deterministic provers. Therefore, we define that the protocol is secure against concurrent attacks and reset attacks if the probability that the adversary wins Game 5 is negligible.

We give an intuitive explanation why Game 5 captures concurrent attacks and reset attacks for challenge-response identification protocols with deterministic provers. Since the prover's response is only determined by the current challenge, it is stateless. Therefore, Experiment$^{id-cr}_{ID,A}(k)$ Execution of protocol ID with adversary $\hat{A}$ and security parameter k in the CR2 setting

```
Initialization:
1. $(pk, sk) \leftarrow ID(\mathsf{keygen}, k)$
2. Choose tape $R_V$ for verifier at random
3. $p \leftarrow 0$

Execute adversary $\hat{A}$ on input $pk$ and reply to its oracle queries as follows:
When $\hat{A}$ makes query $\mathsf{WakeNewProver}$
1. $p \leftarrow p + 1$
2. $SID_p \leftarrow \emptyset$
3. Pick a tape $R_p$ at random
4. Return $p$
When $\hat{A}$ makes query $\mathsf{Send}(\mathsf{prvmsg}, i, msg_1)$
1. $msg_2 \leftarrow ID(\mathsf{prvmsg}, sk, msg_1; R_i)$
2. $sid \leftarrow ID(\mathsf{prvsid}, sk, msg_1; R_i)$,
3. $SID_i \leftarrow SID_i \cup \{sid\}$
4. Return $msg_2 || sid$
When $\hat{A}$ makes query $\mathsf{Send}(\mathsf{vfmsg}, \perp)$
1. $msg_1 \leftarrow ID(\mathsf{vfmsg}, pk, \perp; R_V)$
2. return $msg_1$
When $\hat{A}$ queries $\mathsf{Send}(\mathsf{vfmsg}, msg_1 \parallel msg_2)$ (this query must be after the query $\mathsf{Send}(\mathsf{vfmsg}, \perp)$)
1. $sid_V \parallel decision \leftarrow ID(\mathsf{vfend}, pk, msg_1 \parallel msg_2; R_V)$
2. return $sid_V \parallel decision$

$\hat{A}$ wins the game if the following is true:
1. $decision = accept$ and $sid_V \notin SID_1 \cup \cdots \cup SID_p$
```

Game 2: CR2 for challenge-response identification protocols.

a reset operation does not affect the prover's behaviour, and concurrent execution of the protocol is equivalent to sequential execution of the protocol.

```
Experiment$^{id-cr}_{ID,A}(k)$        Execution of protocol $ID$ with adversary $\hat{A}$ and security
parameter $k$ in the CR2 setting

Initialization:
1. $(pk, sk) \leftarrow ID(\mathsf{keygen}, k)$
2. Choose tape $R_V$ for verifier at random
3. $SID \leftarrow \emptyset$

Execute adversary $\hat{A}$ on input $pk$ and reply to its oracle queries as follows:
When $\hat{A}$ makes query $\mathsf{Send}(\mathsf{prvmsg}, i, msg_1)$
1. $msg_2 \leftarrow ID(\mathsf{prvmsg}, sk, msg_1)$
2. $sid \leftarrow ID(\mathsf{prvsid}, sk, msg_1)$
3. $SID \leftarrow SID \cup \{sid\}$
4. Return $msg_2||sid$
When $\hat{A}$ makes query $\mathsf{Send}(\mathsf{vfmsg}, \perp)$
1. $msg_1 \leftarrow ID(\mathsf{vfmsg}, pk, \perp; R_V)$
2. return $msg_1$
When $\hat{A}$ queries $\mathsf{Send}(\mathsf{vfmsg}, msg_1 \parallel msg_2)$ (this query must be after the query
$\mathsf{Send}(\mathsf{vfmsg}, \perp)$)
1. $sid_V \parallel decision \leftarrow ID(\mathsf{vfend}, pk, msg_1 \parallel msg_2; R_V)$
2. return $sid_V \parallel decision$

$\hat{A}$ wins the game if the following is true:
1. $decision = accept$ and $sid_V \notin SID$
```

Game 3: CR2 for challenge-response identification protocols with deterministic prover.

2.1.2 A Simple Identification Protocol

Description

Initial Setup. The initial setup for our scheme is described in Figure 2.1. We assume the existence of a trusted authority, denoted by $\widehat{TA}$, who will issue certificates for all potential participants in the scheme. Observe that the setup of the scheme is defined in terms of security parameters k' and k. We would probably take $k' = 1024$ and $k = 160$ in practice.

Protocol Description. In a session of the scheme, the prover $\hat{P}$ tries to convince the

C		$\hat{A}$
1. generate (sk, pk)	$\xrightarrow{pk}$	
repeat steps 2. and 3. n_1 times:		
2.	$\xleftarrow{c_i}$	
3.	$\xrightarrow{r_i}$	
4. generate challenge c	$\xrightarrow{c}$	
repeat steps 5. and 6. n_2 times:		
5.	$\xleftarrow{c_i}$	
6.	$\xrightarrow{r_i}$	
7. accept or reject r as the response of c	$\xleftarrow{r}$	
$\hat{A}$ wins the game if C accepts and $c \neq c_i$.		

Game 4: Interactive CR2 for challenge-response identification protocols with deterministic prover.

C		$\hat{A}$
1. generate (sk, pk)	$\xrightarrow{pk}$	
2. generate challenge c	$\xrightarrow{c}$	
repeat steps 3. and 4. $n = n_1 + n_2$ times:		
3.	$\xleftarrow{c_i}$	
4. if $c_i \neq c$, then compute response r_i	$\xrightarrow{r_i}$	
5. accept or reject r as the response of c	$\xleftarrow{r}$	
$\hat{A}$ wins the game if C accepts in step 5.		

Game 5: Deterministic challenge-response identification CR2.

verifier $\hat{V}$ of its identity. $\hat{V}$ "accepts" only if $\hat{P}$ responds to $\hat{V}$'s challenge in an appropriate way. The steps in a session of our scheme are summarized in Figure 2.2.

In the following, we omit the operation "mod p" to simplify the notation. The message flows can be depicted as follows:

$$\hat{P} \quad \xleftarrow{x=g^r} \quad \hat{V}$$
$$\hat{P} \quad \xrightarrow{z=h(x^a)} \quad \hat{V}$$

Input.	Security parameters k and k', which are positive integers. The parameter k' should be polynomial in k.
1.	The $\widehat{TA}$ chooses a large prime p such that $p-1$ is divisible by another large prime q, where $\log_2 p \approx k'$, $\log_2 q \approx k$, and k' is polynomial in k.
2.	The $\widehat{TA}$ chooses an element $g \in \mathbb{Z}_p^*$ having order q.
3.	The $\widehat{TA}$ publishes the triple (p, q, g).
4.	The $\widehat{TA}$ publishes a hash function $h : \mathbb{Z}_p^* \to \{0,1\}^k$.
5.	Each prover $\hat{P}$ chooses a private key a uniformly at random from $\mathbb{Z}_q$, computes the public key $v = g^a \mod p$, and sends v to the $\widehat{TA}$. $\widehat{TA}$ verifies that $\hat{P}$ does possess the private key corresponding to v, and issues a certificate to $\hat{P}$ certifying that v is indeed $\hat{P}$'s public key.

Figure 2.1: Identification scheme setup

1. $\hat{V}$ chooses $r \in \mathbb{Z}_q$ uniformly at random and computes

$$x = g^r \bmod p.$$

Then $\hat{V}$ sends x to $\hat{P}$.

2. After receiving x, $\hat{P}$ rejects and stops if $x^q \bmod p \neq 1$; otherwise $\hat{P}$ computes

$$z = h(x^a \bmod p).$$

and sends z to $\hat{V}$; otherwise $\hat{P}$ rejects and stops.

3. After receiving z, $\hat{V}$ verifies z. If $z = h(v^r \bmod p)$, then $\hat{V}$ accepts; otherwise, $\hat{V}$ rejects.

Figure 2.2: Identification scheme description

Remark. The protocol may have been used in practice. However, to the best of our knowledge, it has not been formally presented or analyzed in the literature. Although having much resemblance, it is not a straightforward instantiation of any conventional encryption or signature based challenge-response identification protocols. Therefore, we think it worthwhile to describe the protocol and analyze its security in detail.

Remark. The protocol can also be implemented in the setting of an elliptic curve E of prime order q, where $q \approx 2^{160}$. In this setting, the verification that $x^q \bmod p \neq 1$ is unnecessary; it would suffice to verify that x is a point on E.

Security proof

Next we prove that the protocol is CR2 secure in the random oracle model if the SDH assumption holds.

Theorem 2.1.1. *If the SDH assumption holds, then the protocol is CR2 secure in the random oracle model.*

Proof. The CR2 security of the protocol is defined by Game 6, which is an instantiation of Game 5. Note that in Game 6, A has access to the hash function $h()$ which is modelled as a random oracle. A may use the hash function to verify if $r_i = h({c_i}^a)$.

C			A
1.	$a \xleftarrow{R} \mathbb{Z}_q, v = g^a$	$\xrightarrow{v}$	
2.	$e \xleftarrow{R} \mathbb{Z}_q, c = g^e$	$\xrightarrow{c}$	
	repeat steps 3. and 4. n times:		
3.		$\xleftarrow{c_i}$	
4.	if ${c_i}^q = 1$ and $c_i \neq c$, then $r_i \leftarrow h({c_i}^a)$	$\xrightarrow{r_i}$	
5.	if $r = h(c^a)$, then accept, else reject	$\xleftarrow{r}$	

Game 6:

We use $\Pr_i[E]$ to indicate the probability that an event E occurs in game i. In Game 6, A wins the game with probability $\Pr_6[r = h(c^a)]$.

Now we transform Game 6 to Game 7 using the following changes:

- C receives g^a and g^e as input where $a \xleftarrow{R} \mathbb{Z}_q$ and $e \xleftarrow{R} \mathbb{Z}_q$, and sets $v = g^a$ and $c = g^e$.
- C is given a restricted DDH oracle $DDH_{g^a}(\cdot, \cdot)$.
- C maintains a table T. The rows of T consists of triples (x, y, h) such that $y = x^a$ and $h = Hash(y)$, where $Hash(\cdot)$ is defined in Algorithm 2 and simulates a random oracle. Let X, Y, and H be the set of the elements in the three columns respectively. Initially, T contains one row (c, y, h_c) where $y = \perp$ ($\perp$ denotes empty) and $h_c \xleftarrow{R} \{0,1\}^k$.
- When A queries the hash function $h()$, C computes the hash value using $Hash()$ in Algorithm 2.

- In step 4, C computes $r_i = Response(c_i)$ using Algorithm 1.
- In Step 5, if $r = h_c$, then C returns the y value in the initial row in T.

$C^{DDH_{g^a}(,)}(g^a, g^e)$			A
1.	$v = g^a$	$\xrightarrow{v}$	
2.	$c = g^e$	$\xrightarrow{c}$	
	repeat steps 3. and 4. n times:		
3.		$\xleftarrow{c_i}$	
4.	if ${c_i}^q =$ and $c_i \neq c$, then $r_i \leftarrow Response(c_i)$	$\xrightarrow{r_i}$	
5.	if $r = h_c$, then return the y value in the initial row (c, y, h_c)	$\xleftarrow{r}$	

Game 7:

```
input: $x$
if $x \in X$ then
    return $h$ in the same row in $T$
end
else
    for each pair $(x, y) \in \{x\} \times Y$ do
        if $DDH_v(x, y) = 1$ then
            fill $x$ in the same row as $y$ in $T$
            return $h$
        end
    end
    add $x$ to a new row of $T$
    fill in a random value for $h$ and $y = \perp$ in that row
    return $h$
end
```

Algorithm 1: Response

In Game 7, C simulates a hash function to answer A's queries to the hash function. C makes consistent replies to identification challenges and hash queries throughout Game 7: when A queries $h(v^{r_i})$, and challenges with g^{r_i}, it receives the same results. Therefore, Game 6 and Game 7 are identical to A, and it holds that

$$\Pr_6[r = h(c^a)] = \Pr_7[r = h(c^a)].$$

```
input: $y$
if $y \in Y$ then
    return $h$ in the same row in $T$
end
else
    for each pair $(x, y) \in X \times \{y\}$ do
        if $DDH_v(x, y) = 1$ then
            fill $y$ in the same row as $x$ in $T$
            return $h$ from the same row
        end
    end
    add $y$ to a new row in $T$
    fill in a random value for $h$ and $x = \perp$ in that row
    return $h$
end
```

Algorithm 2: Hash

Suppose that, in step 5, A responds with $r = h_c$. Then, only with negligible probability can A generate $r = h_c$ without querying the hash function $h(y)$ with $y = c^a$. If A generated r not by querying the hash function, then with negligible probability it holds that $r = h_c$. If A generated r by querying $h(y)$ where $y \neq c^a$, then only with negligible probability it holds that C will return h_c (this only happens in the case that h_c appears in a row which does not contain c). Therefore, if A responds with $r = h_c$, then with overwhelming probability, A queried $h(y)$ where $y = c^a$. It holds that

$$\Pr_7[y = c^a | r = h_c] > 1 - \epsilon$$

where ϵ is negligible. Therefore,

$$\begin{aligned} \Pr_7[y = c^a] &\geq \Pr_7[y = c^a \text{ and } r = h_c] \\ &> (1 - \epsilon) \Pr_7[r = h_c] \\ &= (1 - \epsilon) \Pr_6[r = h(c^a)]. \end{aligned}$$

Therefore, if the SDH assumption holds, then the protocol is CR2 secure. □

We note that the above proof is similar to the proof in [3], where Abdalla *et al.* gave results on the security of an ElGamal encryption variant named DHIES. In both proofs, the key steps are for the challengers C to simulate a hash function so that C's answers to hash queries are consistent with its answers in the protocol.

Equivalence of Protocol Security and SDH Hardness

Next, we prove that the CR2 security of the protocol implies the hardness of the SDH problem. In the proof, we do not need to assume that the hash function $h()$ is a random oracle. Instead, we only assume that $h()$ is collision-free; i.e., the it is computationally infeasible to find x_1 and x_2 where $x_1 \neq x_2$ and $h(x_1) = h(x_2)$. Collision-free is a standard assumption for cryptographic hash functions [102, §4.2].

Theorem 2.1.2. *If the protocol is CR2 secure and the hash function $h()$ is collision-free, then the SDH problem is hard.*

Proof. We prove the theorem by showing that, if the SDH problem is not hard, then the protocol is not CR2 secure or we can find a collision for h.

Suppose that there is a polynomial time algorithm S that solves the SDH problem with non-negligible probability. We define the following Game 8. In Game 8, C and A run a game as defined in Game 6, where A acts as a cheating prover trying to cheat the challenger C. At the same time, A and S run a SDH game where A acts as a challenger to ask S to solve the SDH problem. In the SDH game, A sends $(v = g^a, c = g^e)$ to S, A answers DDH oracle queries $DDH_v(c_i, y_i)$ from S, and S finally answers $y = g^{ae}$.

C		A	S
1. $a \xleftarrow{R} \mathbb{Z}_q, v = g^a$	$\xrightarrow{v}$		
2. $e \xleftarrow{R} \mathbb{Z}_q, c = g^e$	$\xrightarrow{c}$		$\xrightarrow{c,v}$
repeat steps 3. and 4. n times:			
3.		if $c_i = c$ then $d = 2$ else $d = 1$	$\xleftarrow{c_i,y_i}$
	$\xleftarrow{c_i}$	$c_i \leftarrow {c_i}^d$	
4. If ${c_i}^q = 1$ and $c_i \neq c$, then $r_i \leftarrow h({c_i}^a)$	$\xrightarrow{r_i}$	if $r_i = h({y_i}^d)$ then $b = 1$	$\xrightarrow{b}$
		else $b = 0$	
5. If $r = h(v^e)$, then accept, else reject.	$\xleftarrow{r}$	$r = h(y)$	$\xleftarrow{y}$

Game 8:

In the SDH game with S, A uses C to help answer the DDH oracle queries $DDH_v(c_i, y_i)$ from S: if $r_i = h(y_i)$, then A decides that $y_i = {c_i}^a$ and answers with $b = 1$; if $r_i \neq h(y_i)$, then A decides that $y_i \neq {c_i}^a$ and answers with $b = 0$. If it happens that $y_i \neq {c_i}^a$ but $h(y_i) = h({c_i}^a)$, then C finds a collision for $h()$. If this does not happen, then A always gives correct answers to the DDH oracle queries from S. Therefore, in step 5, S will output y such that $y = c^a$.

We conclude that, if the SDH problem is easy, then C will either find a collision for $h()$, or it will accept in Step 5, which means that the protocol is not CR2 secure. $\square$

Combining Theorem 2.1.1 and Theorem 2.1.2, we conclude that the CR2 security of the protocol is equivalent to the hardness of the SDH problem.

Performance

We compare the performance of our scheme with the Stinson-Wu (SW) scheme in [103]. Assume, for all protocols, that $k' = 1024$ and $k = 160$. The total message length is 2208 bits in the SW scheme, and 1184 bits in our scheme. Note however that a hash function can be used to compress the messages in the SW scheme (by hashing the 1024-bit response to 160 bits) so its message length can be reduced to 1344 bits.

In view of the number of exponentiations, the computational complexity of the two schemes is the same. Both the prover and verifier need two exponentiations, all with a 160-bit exponent. In the elliptic curve setting, provers in both schemes are required to perform only one "exponentiation" (i.e., a scalar multiple of a point on the elliptic curve).

2.2 Authenticated Key Exchange Protocols

2.2.1 Security Model

The extended Canetti-Krawczyk (eCK) model is described as an experiment between an adversary $\hat{M}$ and a challenger $\hat{C}$. A certificate authority $\widehat{CA}$ is involved in registering public keys. Initially, $\hat{M}$ selects the identities of n *honest parties*, for whom $\hat{C}$ generates static private key/public key pairs and registers the public keys to $\widehat{CA}$.

Execution of an AKE by one of these parties is called an AKE *session*. A session identifier *sid* is defined as

$$sid = (role, \hat{A}, \hat{B}, comm),$$

where $role = \{I, R\}$ is the role (initiator/responder) of the owner of the session, $\hat{A}$ is the identity of the owner, $\hat{B}$ is the identity of the other party in the session, and *comm* is the concatenation of communication messages between the two parties. Two sessions $sid = (role, \hat{A}, \hat{B}, comm_1)$ and $sid^* = (\overline{role}, \hat{B}, \hat{A}, comm_2)$ are *matching sessions* if $\overline{role}$ is the complement of *role* and $comm_1 = comm_2$. A protocol execution between $\hat{A}$ and $\hat{B}$ without the intervention of an adversary produces two matching sessions.

In the experiment, $\hat{M}$ controls all communications between the parties, and can reveal the static private key of a party, the ephemeral private key in a session, and the session key of a session. $\hat{M}$ can make any sequence of the following queries, which $\hat{C}$ needs to answer accordingly:

- $\mathsf{Send}(\hat{A}, \hat{B}, comm)$. $\hat{M}$ sends a message *comm* to $\hat{A}$ on behalf of $\hat{B}$. $\hat{C}$ returns $\hat{A}$'s response.
- $\mathsf{StaticKeyReveal}(\hat{A})$. $\hat{C}$ returns the static private key of $\hat{A}$.
- $\mathsf{EphemeralKeyReveal}(sid)$. $\hat{C}$ returns the ephemeral private key of the session *sid*.
- $\mathsf{SessionKeyReveal}(sid)$. $\hat{C}$ returns the session key of the session *sid*.
- $\mathsf{Establish}(\hat{A})$. Using this query, the adversary registers an arbitrary public key on behalf of an *adversary controlled party* $\hat{A}$. $\hat{C}$ only checks the validity of the public key, but does not need to check the possession of the corresponding private key.

A session $sid(role, \hat{A}, \hat{B}, comm)$ is *fresh* if the following conditions hold:

- Both $\hat{A}$ and $\hat{B}$ are honest parties.
- $\hat{M}$ did not query the session key of *sid* or its matching session sid^* (if the matching session exists).
- $\hat{M}$ did not query both the static private key of $\hat{A}$ and the ephemeral private key of $\hat{A}$ in this session.
- If sid^* exists, then $\hat{M}$ did not query both the static private key of $\hat{B}$ and the ephemeral private key of $\hat{B}$ in this session.
- If sid^* does not exist, then $\hat{M}$ did not query the static private key of $\hat{B}$.

Security of an AKE is defined as follows. In an eCK experiment, $\hat{M}$ issues Send, StaticKeyReveal, EphemeralKeyReveal, SessionKeyReveal, and Establish queries polynomially many times (in a security parameter λ) in any sequence. Then $\hat{M}$ selects a completed session *sid*, and makes a query $\mathsf{Test}(sid)$. To answer $\mathsf{Test}(sid)$, $\hat{C}$ chooses a bit $b \in \{0, 1\}$ uniformly at random. If $b = 1$, then $\hat{C}$ sets the session key of *sid* as K. Otherwise, $\hat{C}$ selects K from the key space uniformly at random. $\hat{C}$ then returns K as the answer of $\mathsf{Test}(sid)$. $\hat{M}$ continues to query Send, StaticKeyReveal, EphemeralKeyReveal, SessionKeyReveal, and Establish polynomial times. At last, $\hat{M}$ outputs a bit b' and terminates the game. If the selected test session is fresh and $b' = b$, then $\hat{M}$ wins the game.

The advantage of the adversary $\hat{M}$ in the eCK experiment with AKE protocol Π is defined as

$$Adv_{\Pi}^{AKE}(\hat{M}) = \Pr[\hat{M} \text{ wins}] - \frac{1}{2}.$$

An AKE protocol is secure (in the eCK model) if no efficient adversary $\hat{M}$ has more than a negligible advantage in winning the above experiment; i.e.,

$$Adv_{\Pi}^{AKE}(\hat{M}) < 1/Q(\lambda)$$

for any polynomial $Q()$ when λ is sufficiently large.

2.2.2 Efficient Exponentiation Algorithms.

We review several algorithms that will be used in the protocols to accelerate the computations.

Single Exponentiation. To compute a single g^x, a popular algorithm is the *square-and-multiply* algorithm [76, §14.6.1]. Suppose that the bit length of x is t and the cost of a square operation is the same as a multiplication. Then the algorithm on average takes $1.5t$ multiplications to compute g^x.

Multiplication of Exponentiations. To compute the product of k exponentiations ${g_0}^{e_0} \ldots {g_{k-1}}^{e_{k-1}}$, the *simultaneous multiple exponentiation* [76, Algorithm 14.88] can be used to reduce the computation. For $k = 2$, it takes $\frac{7}{4}t+2$ multiplications, for $k = 3$, it takes $\frac{15}{8}t+6$ multiplications, and for $k = 4$, it takes $\frac{31}{16}t + 12$ multiplications. Approximately, it can be estimated that the product of two exponentiations takes 1.17 times a single exponentiation, product of three exponentiations takes 1.25 times a single exponentiation, and product of four exponentiations takes 1.29 times a single exponentiation.

The algorithm requires additional storage. It stores $2^k - 1 - k$ more group elements than computing k separate exponentiations. For small $k = 2, 3, 4$, the overhead is trivial for most applications.

Exponentiations Using the Same Base. To compute two exponentiations g^x and g^y using the same base, the *exponent combination* algorithm [78] can be used. This algorithm takes about 1.17 times the cost of a single exponentiation to compute g^x and g^y simultaneously.

2.2.3 SMEN Protocol

We design the protocol SMEN (Secure MQV or Efficient NAXOS) to achieve tight security reduction (as NAXOS [68]) and efficient computation (as MQV [69]).

Description

In the protocol description, λ is the security parameter. G is a cyclic group of prime order q where $\log_2 q \approx \lambda$. $h_1 : \mathbb{Z}_q \times \mathbb{Z}_q \rightarrow \mathbb{Z}_q$ and $h_2 : G \times G \rightarrow \{0,1\}^\lambda$ are two hash functions. $\hat{A}$ and $\hat{B}$ are two parties with static private/public key pairs $(a, A = g^a)$ and $(b, B = g^b)$ respectively. We assume that the public key certificate of a party can be obtained after knowing its identity. The two-pass SMEN protocol is as follows:

Offline phase

1. $\hat{A}$ selects two ephemeral private keys $\tilde{x_1} \in_R \mathbb{Z}_q, \tilde{x_2} \in_R \mathbb{Z}_q$, and computes

$$x_1 = h_1(\tilde{x_1}, a), X_1 = g^{x_1}, x_2 = h_1(\tilde{x_2}, a), X_2 = g^{x_2}.$$

   $\hat{A}$ stores $\tilde{x_1}, \tilde{x_2}, X_1, X_2$, and erases x_1, x_2.

2. $\hat{B}$ selects two ephemeral private keys $\tilde{y_1} \in_R \mathbb{Z}_q, \tilde{y_2} \in_R \mathbb{Z}_q$, and computes

$$y_1 = h_1(\tilde{y_1}, b), Y_1 = g^{y_1}, y_2 = h_1(\tilde{y_2}, b), Y_2 = g^{y_2}.$$

   $\hat{B}$ stores $\tilde{y_1}, \tilde{y_2}, Y_1, Y_2$, and erases y_1, y_2.

Online phase

1. $\hat{A}$ initializes a session s=(I, $\hat{A}$, $\hat{B}$, X_1, X_2, $\perp$) and sends $(\hat{B}, \hat{A}, X_1, X_2)$ to $\hat{B}$.

2. Upon receiving ($\hat{B}$, $\hat{A}$, X_1, X_2), $\hat{B}$ performs the steps:

   (a) if $\hat{A} = \hat{B}$, then rejects and stops.

   (b) verifies that $X_1 \in G, X_2 \in G$.

   (c) computes $y_1 = h_1(\tilde{y_1}, b), y_2 = h_1(\tilde{y_2}, b)$.

   (d) computes the session key

$$K = h_2(A^{y_1} {X_1}^b {X_2}^{y_2}, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

   (e) erases $\tilde{y_1}, \tilde{y_2}, y_1, y_2$.

(f) completes a session s=(R, $\hat{B}$, $\hat{A}$, X_1, X_2, Y_1, Y_2), and sends ($\hat{A}$, $\hat{B}$, X_1, X_2, Y_1, Y_2) to $\hat{A}$.

3. Upon receiving ($\hat{A}$, $\hat{B}$, X_1, X_2, Y_1, Y_2), $\hat{A}$ performs the following steps:

   (a) if $\hat{B} = \hat{A}$, then rejects and stops.

   (b) verifies that a session s =(I, $\hat{A}$, $\hat{B}$, X_1, X_2, $\perp$) exists.

   (c) verifies that $Y_1 \in G, Y_2 \in G$.

   (d) computes $x_1 = h_1(\tilde{x_1}, a), x_2 = h_1(\tilde{x_2}, a)$.

   (e) computes the session key

   $$K = h_2(B^{x_1} {Y_1}^{a} {Y_2}^{x_2}, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

   (f) erases $\tilde{x_1}, \tilde{x_2}, x_1, x_2$.

   (g) completes the session s =(I, $\hat{A}$, $\hat{B}$, X_1, X_2, Y_1, Y_2).

It is straightforward to verify that, without the intervention of an adversary, $\hat{A}$ and $\hat{B}$ complete with identical shared session keys and matching sessions.

Note that x_1, x_2, y_1, y_2 are deleted in the offline phase and are re-computed in the online phase. This is to prevent the adversary from learning x_1, x_2, y_1, y_2 by revealing a party's ephemeral private key.

Efficiency

In the online phase, each party needs to compute a product of three exponentiations. Using the simultaneous multiple exponentiation algorithm, the cost is about 1.25 exponentiations on average. As a comparison, MQV, HMQV, and CMQV compute a product of two exponentiations in online phase, which takes 1.17 exponentiation.

In the offline phase, each party computes two exponentiations using the same base g. Using the exponent combination algorithm, the cost is 1.17 exponentiations, only 0.17 exponentiation more than that of MQV, HMQV, or CMQV.

Security

Theorem 2.2.1. *SMEN is secure in the eCK model if $h_1()$ and $h_2()$ are modelled as independent random oracles and if the GDH assumption holds.*

Let ϵ_{gdh} be the probability that any polynomial time algorithm solves the GDH problem, and let ϵ_{dl} be the probability that any polynomial time algorithm solves the DL problem. For *any adversary that involves at most n honest parties and activates at most k sessions, we have that*

$$Adv^{AKE}_{SMEN}(\hat{M}) \leq \max\{k^2, nk\}(\epsilon_{gdh} + \epsilon_{dl}).$$

Proof. Define

$$\begin{aligned} & f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2) \\ = \quad & \text{CDH}(A, Y_1)\text{CDH}(B, X_1)\text{CDH}(X_2, Y_2). \end{aligned}$$

Let E indicates a true eCK experiment, let M be the event that $\hat{M}$ wins an eCK experiment, and let H be the event that $\hat{M}$ queried $h_2(f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2), \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ where $sid = (*, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ is the test session. Let $\bar{H}$ be the event that H does not happen. It holds that

$$\begin{aligned} Adv^{AKE}_{SMEN}(\hat{M}) &= \Pr[M|E] - 1/2 \\ &= \Pr[M \wedge H|E] + \Pr[M \wedge \bar{H}|E] - 1/2. \end{aligned}$$

First we consider the event $M \wedge \bar{H}$. Let σ be a 7-tuple

$$\sigma = (f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2), \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

A session key is computed as $K = h_2(\sigma)$. In the protocol, only matching sessions have identical 7-tuples. In the eCK model, $\hat{M}$ is not allowed to reveal the session keys of the test session or its matching session. Since $h_2()$ is modelled as a random oracle, without querying $h_2(\sigma)$ where σ is identical to the 7-tuple of the test session, $\hat{M}$ does not obtain any information about the test session key. It holds that

$$\Pr[M \wedge \bar{H}|E] = 1/2$$

and

$$\begin{aligned} Adv^{AKE}_{SMEN}(\hat{M}) &= \Pr[M \wedge H|E] \\ &\leq \Pr[H|E]. \end{aligned} \tag{2.1}$$

Next we consider the event H. There are two cases that $\hat{M}$ chooses a test session: a test session with a matching session (denoted as event L) and a test session without a matching session (denoted as event $\bar{L}$).

L. The test session has a matching session.

```
if $(\perp, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h) \in T_2$ then
    return h
end
else if $(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h) \in T_2$ where $\alpha = f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ then
    return $h$
end
else
    add $(\perp, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h)$ to $T_2$ where $h \in_R \{0,1\}^\lambda$
    return $h$
end
```

Algorithm 3: $\mathsf{SessionKey}(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$

In this case, we modify the experiment E to E' as follows. In E', $\hat{C}$ selects at random a session sid_A owned by an honest party $\hat{A}$ and a session sid_B owned by an honest party. $\hat{C}$ runs E' the same way as it runs E, except that $\hat{C}$ aborts E' if sid_A and sid_B become non-matching as the experiment proceeds, or sid_A is not chosen by $\hat{M}$ as the test session in the experiment. Let T be the event that $\hat{C}$ does not abort E'; i.e., sid_A is the test session and sid_B is its matching session. It holds that

$$\begin{aligned} \Pr[H|E \wedge L] &= \Pr[H|E' \wedge T] \\ &= \frac{\Pr[H \wedge T|E']}{\Pr[T|E']} \\ &\leq k^2 \Pr[H|E'] \end{aligned} \tag{2.2}$$

We then modify the experiment E' to experiment S in which $\hat{C}$ simulates the hash functions $h_1()$ and $h_2()$. In S, $\hat{C}$ maintains a hash table T_2. T_2 contains tuples $(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h)$ where h is supposed to be the hash value

$$h = h_2(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

T_2 is initially empty. We define two algorithms, $\mathsf{SessionKey}()$ in Algorithm 3 and $\mathsf{Hash}()$ in Algorithm 4, to maintain T_2.

Let (X, Y) be the input of a CDH challenge. We modify E' to S as follows.

- For sid_A, $\hat{C}$ chooses $(\tilde{x_1}, \tilde{x_2})$ and computes X_1 as defined in the protocol, but sets $X_2 = X$, and computes the session key

$$K = \mathsf{SessionKey}(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

```
if $(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h) \in T_2$ then
    return h;
end
else if there is a tuple $(\perp, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2) \in T_2$ and $\alpha = f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$
then
    update the tuple to $(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h)$
    return $h$
end
else
    add $(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h)$ to $T_2$ where $h \in_R \{0,1\}^\lambda$.
    return $h$
end
```

Algorithm 4: $\mathsf{Hash}(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$

If a tuple $(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ already exists in T_2, then SessionKey() needs to check if $\alpha = f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$. Since $a = \text{DLG}(A)$ and $x_1 = \text{DLG}(X_1)$ are known, SessionKey() can check if this relation holds by using the DDH oracle to check if

$$\text{DDH}(X_2, Y_2, \alpha/(B^{x_1} {Y_1}^a)) = 1.$$

- For sid_B, $\hat{C}$ chooses $(\tilde{y_1}, \tilde{y_2})$ and computes Y_1 as defined in the protocol, but sets $Y_2 = Y$, and computes the session key

$$K = \mathsf{SessionKey}(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

If a tuple $(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ already exists in T_2, then SessionKey() needs to check if $\alpha = f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$. Since $b = \text{DLG}(B)$ and $y_1 = \text{DLG}(Y_1)$ are known, SessionKey() can check if this relation holds by using the DDH oracle to check if

$$\text{DDH}(X_2, Y_2, \alpha/(A^{y_1} {X_1}^b)) = 1.$$

- For any other sessions, $\hat{C}$ proceeds according to the protocol faithfully.
- $\hat{C}$ answers the hash query $h_2(\alpha, \hat{I}, \hat{R}, X_1, X_2, Y_1, Y_2)$ by replying with

$$\mathsf{Hash}(\alpha, \hat{I}, \hat{R}, X_1, X_2, Y_1, Y_2).$$

When the query is $h_2(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$, if a tuple $(\perp, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h)$ is already in T_2, Hash() needs to check if $\alpha = f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$. Since each $(\perp, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h) \in T_2$ is added by SessionKey() where $\hat{C}$ knows either (a, x_1) or (b, y_1), Hash() is able to check if the relation holds.

- $\hat{C}$ simulates $h_1()$ in the usual way. When queried with $h_1(x)$, if $h_1(x)$ has not been queried, then $\hat{C}$ returns a random value; otherwise, $\hat{C}$ returns the same value as it returned for $h_1(x)$ before.
- $\hat{C}$ answers the StaticKeyReveal, EphemeralKeyReveal, SessionKeyReveal, and Establish queries faithfully.

The difference between the probabilities that H happens in E' and S is upper bounded by the probability that $\hat{M}$ successfully distinguishes the two experiments. Let D be the output of a distinguisher for the two experiments. It holds that

$$|\Pr[H|E'] - \Pr[H|S]| \leq |\Pr[D = 1|E'] - \Pr[D = 1|S]|.$$

We consider the probability that $\hat{M}$ can distinguish S from E'. The difference between E' and S is due to the fact that $\hat{C}$ does not know $\mathrm{DLG}(X_2)$ or $\mathrm{DLG}(Y_2)$ where $X_2 = X, Y_2 = Y$. However, the session keys involving X_2 or Y_2 are computed by using SessionKey() or Hash(), and SessionKey() and Hash() give consistent results. Since sid_A is the test session, $\hat{M}$ is not allowed to reveal both a and $(\tilde{x_1}, \tilde{x_2})$, or both b and $(\tilde{y_1}, \tilde{y_2})$. Suppose that $\hat{M}$ is able to distinguish E' from S. Then $\hat{M}$ must be able to distinguish at least one of the following four pairs of distributions:

(a) (X, A, a) and $(g^{h_1(a,\tilde{x_2})}, A, a)$ (corresponding to the case that $\hat{M}$ reveals a),

(b) $(X, A, \tilde{x_2})$ and $(g^{h_1(a,\tilde{x_2})}, A, \tilde{x_2})$ (corresponding to the case that $\hat{M}$ reveals $\tilde{x_2}$),

(c) (Y, B, b) and $(g^{h_1(b,\tilde{y_2})}, B, b)$, (corresponding to the case that $\hat{M}$ reveals b),

(d) $(Y, B, \tilde{y_2})$ and $(g^{h_1(b,\tilde{y_2})}, A, \tilde{y_2})$ (corresponding to the case that $\hat{M}$ reveals $\tilde{y_2}$).

Since h_1 is a random oracle, (X, A, a) and $(g^{h_1(a,\tilde{x_2})}, A, a)$ are indistinguishable. If $\hat{M}$ is able to distinguish $(X, A, \tilde{x_2})$ from $(g^{h_1(a,\tilde{x_2})}, A, \tilde{x_2})$, then, in the random oracle model, it can be shown that $\hat{M}$ must have queried $h_1(a, \tilde{x_2})$, therefore, he must have computed $a = \mathrm{DLG}(A)$. Similarly, (Y, B, b) and $(g^{h_1(b,\tilde{y_2})}, B, b)$ are indistinguishable, and if $\hat{M}$ is able to distinguish $(Y, B, \tilde{y_2})$ from $(g^{h_1(b,\tilde{y_2})}, B, \tilde{y_2})$, then $\hat{M}$ must have computed $b = \mathrm{DLG}(B)$. We conclude that the if $\hat{M}$ is able to distinguish E' and S, then $\hat{M}$ is able to solve the DLP. It holds that

$$|\Pr[D = 1|E'] - \Pr[D = 1|S]| \leq \epsilon_{dl}.$$

Therefore,

$$\Pr[H|E'] \leq \Pr[H|S] + \epsilon_{dl} \tag{2.3}$$

If H happens in S, then the inputs to $h_2(f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2), \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ are recorded in T_2. Since $\hat{C}$ knows (a, x_1), $\hat{C}$ can find the record by using the DDH oracle to check if

$$\mathrm{DDH}(X_2, Y_2, \alpha/(B^{x_1} {Y_1}^{a})) = 1,$$

and then find

$$\mathrm{CDH}(X, Y) = \mathrm{CDH}(X_2, Y_2) = \alpha/(B^{x_1} {Y_1}^{a}).$$

In this case, $\hat{C}$ solves a CDH problem using a DDH oracle. It holds that

$$\Pr[H|S] \leq \epsilon_{gdh}. \tag{2.4}$$

Combining (2.2), (2.3), and (2.4), it holds that

$$\Pr[H|E \wedge L] \leq k^2(\epsilon_{gdh} + \epsilon_{dl}). \tag{2.5}$$

$\bar{L}$. The test session does not have a matching session.

In this case, we modify E to E' as follows. $\hat{C}$ randomly chooses an honest party $\hat{B}$ and a session sid_A owned by an honest party $\hat{A}$. $\hat{C}$ runs E' the same way as it runs E, except that $\hat{C}$ aborts E' if the peer in sid_A is not $\hat{B}$, or sid_A is not chosen as the test session in the experiment. Let T be the event $\hat{C}$ does not abort E'; i.e., sid_A is the test session and the peer in this session is B. It holds that

$$\begin{aligned} \Pr[H|E \wedge \bar{L}] &= \Pr[H|E' \wedge T] \\ &= \frac{\Pr[H \wedge T|E']}{\Pr[T|E']} \\ &\leq kn \Pr[H|E'] \end{aligned} \tag{2.6}$$

We then modify the experiment E' to experiment S as follows. Without loss of generality, we assume that $\hat{A}$ is the initiator in the session sid_A.

- For $\hat{B}$, $\hat{C}$ sets $B = Y$. In a session $sid_B = (R, \hat{B}, \hat{O}, X_1, X_2, Y_1, Y_2)$ owned by $\hat{B}$ where B is a responder, $\hat{C}$ picks $(\tilde{y_1}, \tilde{y_2})$ as the ephemeral private keys, chooses $y_1 \in_R \mathbb{Z}_q, y_2 \in_R \mathbb{Z}_q$, and computes $Y_1 = g^{y_1}, Y_2 = g^{y_2}$. $\hat{C}$ computes the session key as

  $$K = \mathsf{SessionKey}(\hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

  If a tuple $(\alpha, \hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2)$ already exists in T_2, then $\mathsf{SessionKey}()$ needs to check if $\alpha = f(\hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2)$. Since y_1 and y_2 are known, $\mathsf{SessionKey}()$ can check if the relation holds by using the DDH oracle to check if

  $$\mathrm{DDH}(B, X_1, \alpha/(O^{y_1} {X_2}^{y_2}) = 1$$

where O is the static public key of the peer $\hat{O}$.

In a session $sid_B = (I, \hat{B}, \hat{O}, X_1, X_2, Y_1, Y_2)$ owned by $\hat{B}$ where B is an initiator, the simulation is similar and we omit the detailed steps.

- For $sid_A = (I, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$, $\hat{C}$ generates $(\tilde{x_1}, \tilde{x_2})$ and computes X_2 according to the protocol, but sets $X_1 = X$. $\hat{C}$ computes the session key as

$$K = \mathsf{SessionKey}(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

  If a tuple $(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ already exists in T_2, then $\mathsf{SessionKey}()$ needs to check if $\alpha = f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$. Since $a = \text{DLG}(A)$ and $x_2 = \text{DLG}(X_2)$ are known, $\mathsf{SessionKey}()$ can check if the relation holds by using the DDH oracle to check if

$$\text{DDH}(X_1, B, \alpha/({Y_1}^a {Y_2}^{x_2})) = 1.$$

- For any other sessions, $\hat{C}$ proceeds according to the protocol faithfully.
- $\hat{C}$ answers the hash query $h_2(\alpha, \hat{I}, \hat{R}, X_1, X_2, Y_1, Y_2)$ by replying with

$$\mathsf{Hash}(\alpha, \hat{I}, \hat{R}, X_1, X_2, Y_1, Y_2).$$

  When the query is $h_2(\alpha, \hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2)$, if a tuple $(\perp, \hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2, h)$ already exists in T_2, $\mathsf{Hash}()$ needs to check if $\alpha = f(\hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2)$. When the query is $h_2(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$, if a tuple $(\perp, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h)$ already exists in T_2, $\mathsf{Hash}()$ needs to check if $\alpha = f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$. Since each $(\perp, \hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2, h) \in T_2$ or $(\perp, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h) \in T_2$ is added by $\mathsf{SessionKey}()$, $\hat{C}$ knows either (y_1, y_2) or (a, x_2) respectively. Therefore, $\mathsf{Hash}()$ is able to check if the relation holds.
- $\hat{C}$ simulates $h_1()$ in a usual way.
- $\hat{C}$ answers the $\mathsf{StaticKeyReveal}$, $\mathsf{EphemeralKeyReveal}$, $\mathsf{SessionKeyReveal}$, and $\mathsf{Establish}$ queries faithfully.

We consider the probability that $\hat{M}$ can distinguish S from E'. The difference between the probabilities that H happens in the two experiments is upper bounded by the probability that $\hat{M}$ successfully distinguishes the two experiments. Let D be output of a distinguisher for the two experiments. It holds that

$$|\Pr[H|E'] - \Pr[H|S]| \leq |\Pr[D = 1|E'] - \Pr[D = 1|S]|.$$

The difference between E' and S is due to the fact that $\hat{C}$ does not know DLG(B) or DLG(X_1) where $X_1 = X, B = Y$. However, the session keys involving B or X_1 are computed by calling SessionKey() or by calling Hash(), and SessionKey() and Hash() give consistent results. Since sid_A is the test session, $\hat{M}$ is not allowed to reveal both a and $(\tilde{x_1}, \tilde{x_2})$, or to reveal DLG$(B)$. In this case, the only way that $\hat{M}$ can distinguish S from E' is if $\hat{M}$ queries $h_1(a, \tilde{x_1})$, $h_1(b, \tilde{y_1})$, or $h_1(b, \tilde{y_2})$ to find out that X_2, Y_1, or Y_2 was not computed correctly. However, $\hat{M}$ cannot do this unless it computes DLG(X_1) or DLG(B) (A more detailed analysis would be similar to the analysis for distinguishing E' and S under the event L). It holds that

$$|\Pr[D = 1|E'] - \Pr[D = 1|S]| \leq \epsilon_{dl}.$$

Therefore,

$$\Pr[H|E'] \leq \Pr[H|S] + \epsilon_{dl} \tag{2.7}$$

If H happens in S, then the inputs of the hash query

$$h_2(f(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)), \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$$

are recorded in T_2. Since $\hat{C}$ knows (a, x_2) for sid_A, $\hat{C}$ can find the record by checking if

$$\text{DDH}(X_1, B, \alpha/({Y_1}^a {Y_2}^{x_2})) = 1,$$

and then find

$$\text{CDH}(X, Y) = \text{CDH}(X_1, B) = \alpha/({Y_1}^a {Y_2}^{x_2})).$$

In this case, $\hat{C}$ solves a CDH problem using a DDH oracle. Then it holds that

$$\Pr[H|S] \leq \epsilon_{gdh}. \tag{2.8}$$

Combining (2.6), (2.7), and (2.8), it holds that

$$\Pr[H|E \wedge \bar{L}] \leq kn(\epsilon_{gdh} + \epsilon_{dl}). \tag{2.9}$$

Combining (2.1), (2.5), and (2.9), we have that

$$\begin{aligned} Adv^{AKE}_{SMEN}(\hat{M}) &\leq \Pr[H|E] \\ &= \Pr[H|E \wedge L]\Pr[L] + \Pr[H|E \wedge \bar{L}]\Pr[\bar{L}] \\ &\leq \max\{\Pr[H|E \wedge L], \Pr[H|E \wedge \bar{L}]\} \\ &= \max\{k^2, kn\}(\epsilon_{gdh} + \epsilon_{dl}). \end{aligned}$$

If the GDH assumption and the DL assumption hold, then ϵ_{gdh} and ϵ_{dl} are negligible in the security parameter λ. Both k and n are polynomial in λ. Therefore, $Adv^{AKE}_{SMEN}(\hat{M})$ is negligible. □

Reflection Attacks

SMEN does not allow a party to establish a key with itself. This is necessary to prevent $\hat{M}$ from distinguishing S from E' under the event $\bar{L}$ (the test session does not have a matching session). Recall that under the event $\bar{L}$, in experiment S, $\hat{B}$'s public key B is replaced with Y. If $\hat{A} = \hat{B}$, then $\hat{M}$ would find out that $A \neq Y$ and deduce that it is in S. From another point of view, if SMEN allows a party to establish a key with itself, then the protocol suffers from the following reflection attack where $\hat{M}$ impersonates $\hat{A}$ to $\hat{A}$: $\hat{M}$ receives $(\hat{A}, \hat{A}, X_1, X_2)$ from $\hat{A}$, chooses y_2, computes $Y_1 = 1/X_1, Y_2 = g^{y_2}$, and sends back $(\hat{A}, \hat{A}, X_1, X_2, Y_1, Y_2)$. The session key is $K = h_2(g^{x_2 y_2}, \hat{A}, \hat{A}, X_1, X_2, Y_1, Y_2)$, and $\hat{M}$ can compute it using y_2.

We note that, although NAXOS and NETS allow a party to establish a key with itself, their security proofs do not cover this case. In the proofs, when the adversary chooses a test session $(role, \hat{A}, \hat{B}, *)$ without a matching session, the simulator changes a true experiment to a simulated experiment by substituting the public key of the peer $\hat{B}$ with V, where V is part of the input of a CDH challenge. The proofs are based on the argument that the two experiments are indistinguishable to the adversary. However, this argument holds only when $\hat{A} \neq \hat{B}$. When $\hat{B} = \hat{A}$, the adversary can find out it is in a simulated experiment because $V \neq A$ where A is $\hat{A}$'s public key. This flaw can be fixed by adding a case to the proof. When $\hat{M}$ chooses $(role, \hat{A}, \hat{A}, *)$ as a test session, the simulator changes the experiment as follows: $\hat{C}$ randomly chooses a party $\hat{A}$, and substitutes its public key with V. $\hat{M}$ chooses $r \in_R \mathbb{Z}_q$ and computes $X = A^r$, and sets the session key as $K \in_R \{0,1\}^\lambda$. If $\hat{M}$ does not choose a session $(role, \hat{A}, \hat{A}, *)$ without a matching session as the test session, then $\hat{C}$ aborts. It can be shown that if $\hat{M}$ wins the experiment, then in the random oracle model, $\hat{C}$ can compute g^{v^2}. If $\hat{C}$ can compute g^{v^2} for given g^v, then it can solve the CDH problem [74].

2.2.4 SMEN$^-$ Protocol

SMEN uses the NAXOS trick to compute an ephemeral public key so that an adversary cannot get its discrete log in the eCK model. This trick needs to use the static private key to compute the ephemeral public key. If we want to minimize the risk of leaking the static private key, then we may try to minimize the use of the static private key. At the same time, we do not want to sacrifice the efficiency or tightness of reduction too much. To achieve this property, we propose SMEN$^-$, which does not use the NAXOS trick, but is still efficient in online computation and tight in security reduction.

Description

In the protocol description, λ is the security parameter. G is a cyclic group of prime order q where $\log_2 q \approx \lambda$. $h : G \times G \to \{0,1\}^{\lambda}$ is a hash function. $\hat{A}$ and $\hat{B}$ are two parties with static public/private key pairs $((A_1, A_2), (a_1, a_2))$ and $((B_1, B_2), (b_1, b_2))$ respectively, where $A_1 = g^{a_1}, A_2 = g^{a_2}, B_1 = g^{b_1}, B_2 = g^{b_2}$. We assume that the public key certificate of a party can be obtained after knowing its identity. The two-pass SMEN$^-$ protocol is as follows:

Offline phase

1. $\hat{A}$ selects $x_1 \in_R \mathbb{Z}_q, x_2 \in_R \mathbb{Z}_q$, computes $X_1 = g^{x_1}, X_2 = g^{x_2}$, and stores x_1, x_2, X_1, X_2.
2. $\hat{B}$ selects $y_1 \in_R \mathbb{Z}_q, y_2 \in_R \mathbb{Z}_q$, computes $Y_1 = g^{y_1}, Y_2 = g^{y_2}$, and stores y_1, y_2, Y_1, Y_2.

Online phase

1. $\hat{A}$ initializes a session s=(I, $\hat{A}$, $\hat{B}$, X_1, X_2, $\perp$) and sends $(\hat{B}, \hat{A}, X_1, X_2)$ to $\hat{B}$.
2. Upon receiving ($\hat{B}$, $\hat{A}$, X_1, X_2), $\hat{B}$ performs the steps:
   (a) if $\hat{A} = \hat{B}$, then $\hat{B}$ rejects and stops.
   (b) verifies that $X_1 \in G$ and $X_2 \in G$.
   (c) computes the session key $K = h({A_1}^{y_1}{X_1}^{b_1}{A_2}^{b_2}{X_2}^{y_2}, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$.
   (d) sends ($\hat{A}$, $\hat{B}$, X_1, X_2, Y_1, Y_2) to $\hat{A}$.
   (e) completes session s=(R, $\hat{B}$, $\hat{A}$, X_1, X_2, Y_1, Y_2).
3. Upon receiving ($\hat{A}$, $\hat{B}$, X_1, X_2, Y_1, Y_2), $\hat{A}$ performs the following steps:
   (a) if $\hat{A} = \hat{B}$, then $\hat{A}$ rejects and stops.
   (b) verifies that a session s =(I, $\hat{A}$, $\hat{B}$, X_1, X_2, $\perp$) exists.
   (c) verifies that $Y_1 \in G$ and $Y_2 \in G$.
   (d) computes the session key $K = h({B_1}^{x_1}{Y_1}^{a_1}{B_2}^{a_2}{Y_2}^{x_2}, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$.
   (e) completes session s =(I, $\hat{A}$, $\hat{B}$, X_1, X_2, Y_1, Y_2).

It is straightforward to verify that, without the intervention of an adversary, $\hat{A}$ and $\hat{B}$ complete with identical shared session keys and matching sessions.

Efficiency

In the online phase, each party needs to compute a product of four exponentiations. Using the simultaneous multiple exponentiation algorithm, the cost is about 1.29 exponentiations on average.

Security

Theorem 2.2.2. *SMEN$^-$ is secure in the eCK model if the hash function $h()$ is modelled as a random oracle and if the GDH assumption holds.*

Let ϵ_{gdh} be the probability that any polynomial time algorithm solves the GDH problem. For any adversary that involves at most n honest parties and activates at most k sessions, we have that

$$Adv^{AKE}_{SMEN^-}(\hat{M}) \leq \max\{k^2, nk\}\epsilon_{gdh}.$$

Proof. Define

$$\begin{aligned} & f_2(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2) \\ = \;& \mathrm{CDH}(A_1, Y_1)\mathrm{CDH}(B_1, X_1)\mathrm{CDH}(A_2, B_2)\mathrm{CDH}(X_2, Y_2). \end{aligned}$$

Let E indicates a true eCK experiment, let M be the event that $\hat{M}$ wins an eCK experiment, and let H be the event that $\hat{M}$ queried $h(f_2(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2), \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ where $sid = (*, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ is the test session. As with SMEN, it holds that

$$Adv^{AKE}_{SMEN^-}(\hat{M}) \;\leq\; \Pr[H|E]. \tag{2.10}$$

In the eCK model, $\hat{M}$ is not allowed to reveal both the static private key and the ephemeral private key of a test session. If the test session has a matching session, then $\hat{M}$ is not allowed to reveal both the static private key and the ephemeral private key of the matching session, either. If the test session does not have a matching session, then $\hat{M}$ is not allowed to reveal the static private key of the peer in the session. There are six cases where $\hat{M}$ chooses a test session. We denote them as $L_1, \ldots, L_6$, and describe the simulation and analysis for each case as follows. Let (X, Y) be the input of a CDH challenge.

L_1. The test session does not have a matching session, and $\hat{M}$ does not reveal the ephemeral private key of the owner of the test session.

In this case, we modify E to E' as follows. $\hat{C}$ chooses an honest party $\hat{B}$ and a session sid_A owned by an honest party $\hat{A}$ at random. $\hat{C}$ runs E' the same way as it runs E, except that $\hat{C}$ aborts E' if the peer in sid_A is not $\hat{B}$, or sid_A is not chosen as the test session in the experiment. Let T be the event $\hat{C}$ does not abort E'. It holds that

$$\begin{aligned} \Pr[H|E \wedge \bar{L}] &= \Pr[H|E' \wedge T] \\ &= \frac{\Pr[H \wedge T|E']}{\Pr[T|E']} \\ &\leq kn \Pr[H|E'] \end{aligned} \tag{2.11}$$

We then modify the experiment E' to experiment S. In S, $\hat{C}$ simulates $h()$ the same way as $\hat{C}$ simulates $h_2()$ in SMEN, except that the function $f()$ in SMEN is substituted with $f_2()$. Without loss of generality, we assume that $\hat{A}$ is an initiator in a session in which it participates, and $\hat{B}$ is a responder in a session in which it participates.

- For $\hat{B}$, $\hat{C}$ sets $B_1 = Y$. In a session $sid_B = (R, \hat{B}, \hat{O}, X_1, X_2, Y_1, Y_2)$ owned by $\hat{B}$, $\hat{C}$ picks $(\tilde{y_1}, \tilde{y_2})$ as the ephemeral private keys, chooses $y_1 \in_R \mathbb{Z}_q, y_2 \in_R \mathbb{Z}_q$, and computes $Y_1 = g^{y_1}, Y_2 = g^{y_2}$. $\hat{C}$ computes the session key as

  $$K = \mathsf{SessionKey}(\hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

  If a tuple $(\alpha, \hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2)$ is already in T_2, then $\mathsf{SessionKey}()$ needs to check if $\alpha = f_2(\hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2)$. Since b_2, y_1 and y_2 are known, $\mathsf{SessionKey}()$ can check if the relation holds by using the DDH oracle to check if

  $$\text{DDH}(B_1, X_1, \alpha/({O_1}^{y_1}{O_2}^{b_2}{X_2}^{y_2}) = 1$$

  where (O_1, O_2) is the public static key of the peer $\hat{O}$.

- For $sid_A = (I, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$, $\hat{C}$ generates $(\tilde{x_1}, \tilde{x_2})$ and computes X_2 according to the protocol, but sets $X_1 = X$. $\hat{C}$ computes the session key as

  $$K = \mathsf{SessionKey}(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

  If a tuple $(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$ already exists in T_2, then $\mathsf{SessionKey}()$ needs to check if $\alpha = f_2(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$. Since a_1, a_2 and x_2 are known, $\mathsf{SessionKey}()$ can check if the relation holds by using the DDH oracle to check if

  $$\text{DDH}(X_1, B_1, \alpha/({Y_1}^{a_1}{B_2}^{a_2}{Y_2}^{x_2})) = 1.$$

- For any other sessions, $\hat{C}$ proceeds according to the protocol faithfully.

- $\hat{C}$ answers the hash query $h_2(\alpha, \hat{I}, \hat{R}, X_1, X_2, Y_1, Y_2)$ by replying with

  $$\mathsf{Hash}(\alpha, \hat{I}, \hat{R}, X_1, X_2, Y_1, Y_2).$$

  When the query is $h_2(\alpha, \hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2)$, if a tuple $(\perp, \hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2, h)$ already exists in T_2, $\mathsf{Hash}()$ needs to check if

  $$\alpha = f_2(\hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

  When the query is $h_2(\alpha, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$, if a tuple $(\perp, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h)$ already exists in T_2, then $\mathsf{Hash}()$ needs to check if

  $$\alpha = f_2(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2).$$

  Since each $(\perp, \hat{O}, \hat{B}, X_1, X_2, Y_1, Y_2, h) \in T_2$ or $(\perp, \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2, h) \in T_2$ is added by $\mathsf{SessionKey}()$, $\hat{C}$ knows either (b_2, y_1, y_2) or (a_1, a_2, x_2) respectively. Therefore, $\mathsf{Hash}()$ is able to check if the relation holds.

- $\hat{C}$ simulates $h_1()$ in a usual way.

- $\hat{C}$ answers the $\mathsf{StaticKeyReveal}$, $\mathsf{EphemeralKeyReveal}$, $\mathsf{SessionKeyReveal}$, and $\mathsf{Establish}$ queries faithfully.

The difference between E' and S is due to the fact that $\hat{C}$ does not know $\mathrm{DLG}(B_1)$ or $\mathrm{DLG}(X_1)$ where $X_1 = X, B_1 = Y$. However, the session keys involving B_1 or X_1 are computed by $\mathsf{SessionKey}()$ or by $\mathsf{Hash}()$, and $\mathsf{SessionKey}()$ and $\mathsf{Hash}()$ give consistent results. In event L_1, since $\hat{M}$ does not reveal the ephemeral private key of $\hat{A}$ or static private key of $\hat{B}$, E' and S are identical to $\hat{M}$. Therefore,

$$\Pr[H|E'] = \Pr[H|S]. \tag{2.12}$$

If H happens in S, then the inputs of the hash query

$$h_2(f_2(\hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)), \hat{A}, \hat{B}, X_1, X_2, Y_1, Y_2)$$

is recorded in T_2. Since $\hat{C}$ knows (a_1, a_2, x_2) for sid_A, $\hat{C}$ can find the record by checking if

$$\mathrm{DDH}(X_1, B_1, \alpha/({Y_1}^{a_1}{B_2}^{a_2}{Y_2}^{x_2})) = 1,$$

and then find

$$\mathrm{CDH}(X, Y) = \mathrm{CDH}(X_1, B_1) = \alpha/({Y_1}^{a_1}{B_2}^{a_2}{Y_2}^{x_2})).$$

In this case, $\hat{C}$ solves a CDH problem using a DDH oracle. Then it holds that

$$\Pr[H|S] \leq \epsilon_{gdh}. \tag{2.13}$$

Combining (2.11), (2.12), and (2.13), it holds that

$$\Pr[H|E \wedge L_1] \leq kn\epsilon_{gdh}.$$

For the following cases, we only describe the simulation and the analysis result. It is straightforward to work out the details following the same approach in L_1.

L_2 $\hat{M}$ chooses a test session without a matching session. $\hat{M}$ does not reveal the static private key of the session owner.

Change E to E': $\hat{C}$ chooses a session sid_A at random. Let $\hat{A}$ be the owner of sid_A. $\hat{C}$ chooses a party $\hat{B}$ at random. $\hat{C}$ stops if the peer in sid_A is not $\hat{B}$, or sid_A is not the test session. It holds that

$$\Pr[H|E \wedge L_2] = nk \Pr[H|E'].$$

Change E' to S: $\hat{C}$ sets $A_2 = X$, $B_2 = Y$.

We have that

$$\Pr[H|E \wedge L_2] \leq nk\epsilon_{gdh}.$$

L_3. $\hat{M}$ chooses a test session with a matching session. $\hat{M}$ does not reveal the static private key of the session owner. $\hat{M}$ does not reveal the static private key of the matching session owner.

Change E to E': $\hat{C}$ chooses a session sid_A at random. Let $\hat{A}$ be the owner of the session. $\hat{C}$ chooses another party $\hat{B}$ at random. $\hat{C}$ stops if sid_A is not the test session or the other party in sid_A is not $\hat{B}$. It holds that

$$\Pr[H|E \wedge L_3] = nk \Pr[H|E'].$$

Change E' to S: $\hat{C}$ sets $A_2 = X$, $B_2 = Y$.

We have that

$$\Pr[H|E \wedge L_3] \leq nk\epsilon_{gdh}.$$

L_4. $\hat{M}$ chooses a test session with a matching session. $\hat{M}$ does not reveal the ephemeral private key of the session owner. $\hat{M}$ does not reveal the static private key of the matching session owner.

Change E to E': $\hat{C}$ chooses a session sid_A at random. Let $\hat{A}$ be the owner of sid_A. $\hat{C}$ chooses another party $\hat{B}$ at random. $\hat{C}$ stops if the peer in sid_A is not $\hat{B}$, or sid_A is not the test session. It holds that

$$\Pr[H|E \wedge L_4] = nk \Pr[H|E'].$$

Change E' to S: $\hat{C}$ sets $X_1 = X$, $B_1 = Y$.

We have that

$$\Pr[H|E \wedge L_4] \leq nk\epsilon_{gdh}.$$

L_5. $\hat{M}$ chooses a test session with a matching session. $\hat{M}$ does not reveal the static private key of the session owner. $\hat{M}$ does not reveal the ephemeral private key of the matching session owner.

Change E to E': $\hat{C}$ chooses a party $\hat{A}$ at random and chooses a session sid_B at random. Let $\hat{B}$ be the owner of sid_B. $\hat{C}$ stops if the matching session of sid_B is not the test session, or the the peer in sid_B is not $\hat{A}$. It holds that

$$\Pr[H|E \wedge L_5] = nk \Pr[H|E'].$$

Change E' to S: $\hat{C}$ sets $A_1 = X$, $Y_1 = Y$.

We have that

$$\Pr[H|E \wedge L_5] \leq nk\epsilon_{gdh}.$$

L_6. $\hat{M}$ chooses a test session with a matching session. $\hat{M}$ does not reveal the ephemeral private key of the session owner. $\hat{M}$ does not reveal the ephemeral private key of the matching session owner.

Change E to E': $\hat{C}$ chooses a session sid_A at random. Let $\hat{A}$ be the owner of sid_A. $\hat{C}$ chooses a session sid_B at random. Let $\hat{B}$ be the owner of sid_B. $\hat{C}$ stops if sid_A and sid_B are not matching or sid_A is not the test session. It holds that

$$\Pr[H|E \wedge L_6] = k^2 \Pr[H|E'].$$

Change E' to S: $\hat{C}$ sets $X_2 = X$, $Y_2 = Y$.

We have that

$$\Pr[H|E \wedge L_6] \leq k^2\epsilon_{gdh}.$$

Summarizing the above results, it holds that

$$\begin{aligned} Adv^{AKE}_{SMEN^-}(\hat{M}) &\leq \Pr[H|E] \\ &\leq \sum_{i=1}^{6} \Pr[H|E \wedge L_i]\Pr[L_i] \\ &\leq \max\{\Pr[H|E \wedge L_i], 1 \leq i \leq 6\} \\ &= \max\{k^2, kn\}\epsilon_{gdh}. \end{aligned}$$

□

2.2.5 Comparison

In Table 2.1, we compare the efficiency and security of MQV, HMQV, CMQV, NAXOS, NETS, SMEN and SMEN$^-$. We assume that, in all the protocols, 1) it is not necessary to check if $x^q = 1$ to verify whether x is an element of a group G of order q, and 2) efficient algorithms are used to compute the product of multiple exponentiations.

Protocol	Efficiency		Security	Assumption	Proof	NAXOS
	Offline	Online				trick
MQV	1	1.17	KKS, wPFS, KCI, UKS	?	?	No
HMQV	1	1.17	CK, wPFS, KCI, LEP	ROM, GDH, KEA	not tight	No
CMQV	1	1.17	eCK	ROM, GDH	not tight	Yes
NAXOS	1	2.17	eCK	ROM, GDH	tight	Yes
NETS	1	2	eCK	ROM, GDH	tight	Yes
SMEN$^-$	1.17	1.29	eCK	ROM, GDH	tight	No
SMEN	1.17	1.25	eCK	ROM, GDH	tight	Yes

Table 2.1: Efficiency (in number of exponentiations) and security comparison.

2.3 ElGamal Encryption and Variants

2.3.1 Security of ElGamal Encryption

Scheme Description

First we recall the basic ElGamal encryption scheme. Let G be a multiplicative group of prime order q and g be a generator of G. $k \approx \log_2 q$ will be used as the security parameter in the security analysis. The scheme consists of three algorithms: key generation, encryption, and decryption. G, g, q are default system parameters for these algorithms. In the following description, we use $x \in_R X$ to indicate that x is chosen from set X uniformly at random.

The key generation algorithm computes a public key u and a private key a as follows:

$$a \in_R \mathbb{Z}_q, u \leftarrow g^a.$$

The message space of the scheme is G. To encrypt a message $m \in G$, the encryption algorithm computes a ciphertext $c = (x, y) \in G \times G$ as follows:

$$r \in_R \mathbb{Z}_q, x \leftarrow g^r, y \leftarrow m \cdot u^r.$$

To decrypt a ciphertext $c = (x, y) \in G \times G$, the decryption algorithm computes

$$m \leftarrow y/x^a.$$

Security Analysis

First we present the Strong GKEA (SGKEA) and DTDLA.

Assumption 2.3.1. *The* Strong Generalized Knowledge-of-Exponent Assumption (SGKEA) *is as follows: Let G be a group of prime order q, g be a generator of G, and $k \approx \log_2 q$ be the security parameter. Let A be a polynomial time (in k) algorithm. A is given $(x_0, {x_0}^a, \cdots, x_n, {x_n}^a)$ where $x_0, \cdots, x_n \in G$ and $x_1, \cdots, x_n$ are chosen by A adaptively, n is polynomial in k, $a \in_R \mathbb{Z}_q$ and a is unknown to A. There exists an efficient compiler E such that for any A that outputs a pair $(x, y) \in G \times G$, E can compiles A to A' that satisfies the following conditions: 1. $\hat{A}'$ is polynomial time; 2. A' has the same input, output, and random tape accesses as A, except that in addition to x and y, A' also outputs $(c_0, \cdots, c_n)$ such that*

$$\Pr\left[\prod_{i=0}^{n} {x_i}^{c_i} = x \mid y = x^a\right] > 1 - \epsilon_{sgkea}$$

where ϵ_{sgkea} is negligible.

SGKEA is a variant of GKEA. GKEA was first defined in [48]. GKEA is the same as SGKEA except that it does not specify if A chooses $x_1, \cdots, x_n$.

Assumption 2.3.2. *The* Delayed-Target Discrete Log Assumption (DTDLA) *is as follows. Let G be a finite cyclic group, g be a generator of G, and $k \approx \log_2 |G|$. Let A be a probabilistic polynomial (in k) time algorithm that takes input g and has access to two oracles. The first* *is a discrete log oracle $DL_g()$, which on input $x \in G$ returns r such that $x = g^r$. The second is a challenge oracle $C_g()$ that, when invoked, returns $x \in_R G$. A can access $DL_g()$ n times, where n is polynomial in k. The DTDLA assumption assumes that after receiving a challenge x from $C_g()$, without further accesses to the oracle $DL_g()$, the probability that A outputs r such that $g^r = x$ is negligible.*

DTDLA is defined and discussed in [64].

Next we review the security notation. We define the following interactive game, Game 9, between a probabilistic polynomial time (PPT) challenger C and a PPT adversary A. In the game, A can ask for n decryptions from C. Then A tries to decrypt a fresh challenge ciphertext.

C		A
1.	$a \in_R \mathbb{Z}_q, u \leftarrow g^a$	$\xrightarrow{g, u}$
	Repeat 2 and 3 n times:	
2.		$\xleftarrow{x_i, y_i}$
3.	$m_i \leftarrow y_i/{x_i}^a$	$\xrightarrow{m_i}$
4.	$m \in_R G, r \in_R \mathbb{Z}_q, x \leftarrow g^r, y \leftarrow m \cdot u^r$	$\xrightarrow{x, y}$
5.		$\xleftarrow{m'}$

Game 9: ElGamal OW-CCA1 Game.

We use $\Pr_i[e]$ to denote the probability that an event e happens in Game i. We say that ElGamal encryption is one-way under non-adaptive chosen ciphertext attack (i.e., OW-CCA1 secure) if $\Pr_9[m' = m]$ is negligible.

We show that ElGamal encryption is OW-CCA1 secure if SGKEA and DTDLA hold. The sketch of the proof is as follows: assuming that SGKEA holds, if an adversary can break the scheme, then using the adversary as a subroutine, a PPT algorithm can break the DTDLA. We follow the proof style suggested in [96] to structure the proof as a sequence of games.

Theorem 2.3.3. *If SGKEA and DTDLA hold, then the ElGamal encryption scheme is OW-CCA1 secure.*

Proof. We transform Game 9 to Game 10 by removing the value m in the messages. It is clear that

$$\Pr_9[m' = m] = \Pr_{10}[z = x^a]. \tag{2.14}$$

	C		A
1.	$a \in_R \mathbb{Z}_q, u \leftarrow g^a$	$\xrightarrow{g, u}$	
	Repeat 2 and 3 n times:		
2.		$\xleftarrow{x_i}$	
3.	$z_i \leftarrow {x_i}^a$	$\xrightarrow{z_i}$	
4.	$x \in_R G$	$\xrightarrow{x}$	
5.		$\xleftarrow{z}$	

Game 10:

We transform Game 10 to Game 11 by a conceptual change: instead of receiving x from C, A reads x from its random tape. Besides, A outputs x along with z. It holds that

$$\Pr_{11}[z = x^a] = \Pr_{10}[z = x^a]. \tag{2.15}$$

	C		A
1.	$a \in_R \mathbb{Z}_q, u \leftarrow g^a$	$\xrightarrow{g, u}$	
	Repeat 2 and 3 n times:		
2.		$\xleftarrow{x_i}$	
3.	$z_i \leftarrow {x_i}^a$	$\xrightarrow{z_i}$	
4.			$x \in_R G$
5.		$\xleftarrow{x, z}$	

Game 11:

In Game 11, when A outputs the pair (x, z), by SGKEA, it can be compiled into A' that has the same input, output, and random tape accesses as A, except that, in addition to (x, z), A' also outputs $r_0, \cdots, r_n$, such that

$$\Pr\left[x = \prod_{0 \le i \le n} {x_i}^{r_i} | z = x^a\right] > 1 - \epsilon_{sgkea}. \tag{2.16}$$

Next we transform Game 11 to a new Game 12 by replacing A with A'.

Next we transform Game 12 to a new Game 13. In Game 13, C can query $DL_g(x)$ to compute the logarithm of x, and A' queries $C_g()$ to generate a random x. The oracles $DL_g()$ and $C_g()$ are as defined in the DTDLA assumption.

C			A'
1.	$a \in_R \mathbb{Z}_q, u \leftarrow g^a$	$\xrightarrow{g, u}$	
	Repeat 2 and 3 n times:		
2.		$\xleftarrow{x_i}$	
3.	$z_i \leftarrow {x_i}^a$	$\xrightarrow{z_i}$	
4.			$x \in_R G$
5.		$\xleftarrow{x, z, r_0, \cdots, r_n}$	

Game 12:

C			A'
1.	$a \in_R \mathbb{Z}_q, u \leftarrow g^a$	$\xrightarrow{g, u}$	
	Repeat 2 - 4 n times:		
2.		$\xleftarrow{x_i}$	
3.	$e_i \leftarrow DL_g(x_i)$		
4.	$z_i \leftarrow {x_i}^a$	$\xrightarrow{z_i}$	
5.			$x \leftarrow C_g$
6.		$\xleftarrow{x, z, r_0, \cdots, r_n}$	
7.	$e = r_0 + e_1 r_1 + e_2 r_2 + \cdots + e_n r_n$		

Game 13:

It is clear that

$$\Pr_{13}\left[g^e = x\right] = \Pr_{13}\left[x = \prod_{0 \le i \le n} {x_i}^{r_i}\right] \tag{2.17}$$

$$= \Pr_{11}\left[x = \prod_{0 \le i \le n} {x_i}^{r_i}\right]. \tag{2.18}$$

In Game 13, $DL_g()$ is accessed n times, then $C_g()$ outputs a random challenge x. e is computed as a "guess" of the logarithm of x. Therefore,

$$\Pr_{13}\left[g^e = x\right] = Adv_{dtdl}^{C} \tag{2.19}$$

where Adv_{dtdl}^{C} is the probability that the polynomial time algorithm C can solve the delayed-target DL problem.

Combining (2.14)-(2.19), it holds that

$$\Pr_9[m' = m] < \frac{Adv_{dtdl}^{C}}{1 - \epsilon_{sgkea}}.$$

We conclude that if SGKEA and DTDLA hold, then ElGamal is OW-CCA1 secure. □

Relations Between The Assumptions

First, we consider the relation between OW-CCA1 security of ElGamal encryption and the following delayed-target computational Diffie-Hellman assumption (DTCDHA). DTCDHA is defined in [46] and is discussed in [64].

Assumption 2.3.4. *The* Delayed-Target Computational Diffie-Hellman Assumption (DTCDHA) *is as follows. Let G be a finite cyclic group of order q, g be a generator of G, and $k \approx \log_2 q$. Let A be a probabilistic polynomial (in k) time algorithm that takes input $g, g^a \in G$ where $a \in_R \mathbb{Z}_q$. A has access to two oracles. The first is a CDH oracle $CDH_{g,g^a}()$, which on input $x \in G$ returns x^a. The second is a challenge oracle $C_g()$ that, when invoked, returns $x \in_R G$. A can access $CDH_{g,g^a}()$ n times where n is polynomial in k. The DTCDHA assumes that after receiving a challenge x from $C_g()$, without further access to the oracle $CDH_{g,g^a}()$, the probability that A outputs z such that $z = x^a$ is negligible.*

We observe that Game 10 is in fact a delayed-target computational Diffie-Hellman game. Therefore we have the result:

Theorem 2.3.5. *OW-CCA1 security of ElGamal encryption is equivalent to DTCDHA.*

Since DTDLA and SGKEA imply that the ElGamal encryption is OW-CCA1 secure, it also holds that

Corollary 2.3.6. *DTDLA and SGKEA imply DTCDHA.*

This result may be of independent interest in studying the relation between the assumptions.

In [21], Brown and Gallant presented an algorithm that can be used to recover a in the DTCDH problem. If the adversary knows u where $u|(q-1)$ and $u \approx q^{1/3}$, then the adversary can query the CDH oracle $\Theta(q^{1/3})$ times and recover a in time $\Theta(q^{1/3})$. This algorithm is more efficient than Pallard's ρ algorithm [76, §3.6], which solves a in time $O(q^{1/2})$ without querying the oracle.

2.3.2 Security of Damgård ElGamal Encryption

In this section, we use the delayed-target decisional Diffie-Hellman assumption (DTDDHA), which is the Gap Subgroup Membership Assumption in prime order groups, to prove that

DEG is IND-CCA1. Our proof is simpler than the one in [49] in that it uses a straightforward reduction. Then, we propose a decisional version of the DHK1 (Diffie-Hellman Knowledge) assumption [9], namely the DDHK1 assumption, prove that DEG is IND-CCA1 under the DDHK1 and DDH assumptions, and prove that DHK1 implies DDHK1. In [9], DHK1 is used to prove the security of DEG. Our result shows that DHK1 is stronger than necessary in the security proof of DEG.

Scheme Description

Let G be a group of prime order q and let g be a generator of G. DEG consists of three algorithms: key generation, encryption, and decryption. G, g, q are default system parameters for these algorithms.

The key generation algorithm computes a public key $(u, v) \in G \times G$ and a private key $(a, b) \in \mathbb{Z}_q \times \mathbb{Z}_q$ as follows:

$$a \in_R \mathbb{Z}_q, b \in_R \mathbb{Z}_q, u \leftarrow g^a, v \leftarrow g^b.$$

The message space of the scheme is G. To encrypt a message $m \in G$, the encryption algorithm computes a ciphertext $c = (x, y, z) \in G^3$ as follows:

$$r \in_R \mathbb{Z}_q, x \leftarrow g^r, y \leftarrow u^r, z \leftarrow m \cdot v^r.$$

To decrypt a ciphertext c, the decryption algorithm computes m as follows: if $y = x^a$, then

$$m \leftarrow z/x^b.$$

Otherwise, the decryption algorithm returns $\perp$ to indicate an invalid ciphertext.

Security Analysis

First we define the DTDDHA:

Assumption 2.3.7. *The* Delayed-Target Decisional Diffie-Hellman Assumption (DTDDHA) *is as follows: Let G be a group of prime order q, g be a generator of G, and $k \approx \log_2 q$. Let D be a probabilistic polynomial (in k) time algorithm that takes input $g, g^a \in G$ where $a \in_R \mathbb{Z}_q$ and A has access to two oracles. The first is a DDH oracle $DDH_{g,g^a}()$, which on input $(x, y) \in G \times G$ returns 1 if $y = x^a$ and returns 0 otherwise. The second is a challenge oracle $C_{g,g^a}()$ that, when invoked, returns a challenge (x, x^a) or (x, y) with equal probability where $x \in_R G$ and $y \in_R G$. A can access $DDH_{g,g^a}()$ for n times where n is polynomial in k.*

The DTDDHA assumes that after receiving a challenge (x, y) from $C_{g,g^a}()$, without further accesses to the oracle $DDH_{g,g^a}()$, the advantage of D in this game, defined as

$$Adv^D_{dtddh} = \left|\Pr\left[D(x, y) = 1 | y = x^a\right] - \Pr\left[D(x, y) = 1 | y \in_R G\right]\right|,$$

is negligible.

DTDDHA is an instantiation of the Gap Subgroup Membership Assumption in [49].

Remark. It holds that DDHA implies CDHA, which implies DLA in turn. This relation cannot be shown to hold for DTDDHA, DTCDHA, and DTDLA. Although the task for the adversary in DTDDHA is easier than in DTCDHA, the oracle it has in DTDDHA is weaker than that in DTCDHA. Similarly, although the task for the adversary in DTCDHA is easier than in DTDLA, the oracle it has in DTCDHA is weaker than that in DTDLA.

Next we describe an interactive game, Game 14, between a PPT challenger C and a PPT adversary A to define the semantic security of DEG under CCA1.

C	A
1. $a \in_R \mathbb{Z}_q, b \in_R \mathbb{Z}_q, u \leftarrow g^a, v \leftarrow g^b$	$\xrightarrow{g, u, v}$
Repeat 2 - 5 n times:	
2.	$\xleftarrow{x_i, y_i, z_i}$
3. if $y_i = {x_i}^a$ then $m_i \leftarrow z_i / {x_i}^b$	
4. else $m_i = \perp$	
5.	$\xrightarrow{m_i}$
6.	$\xleftarrow{m'_0, m'_1}$
7. $d \in_R \{0, 1\}, r \in_R \mathbb{Z}_q, x \leftarrow g^r$	
$y \leftarrow u^r, z' \leftarrow v^r, z \leftarrow m'_d z'$	$\xrightarrow{x, y, z}$
8.	$\xleftarrow{d'}$

Game 14: DEG CCA1 Game.

The adversary's advantage in Game 14 is

$$Adv^A_{game14} = \left|\Pr_{14}\left[d' = d\right] - 1/2\right|.$$

We say that DEG is IND-CCA1 secure if Adv^A_{game14} is negligible.

Next we prove the result:

Theorem 2.3.8. *DEG is IND-CCA1 secure if DTDDHA holds.*

```
$C$                                                          $A$
1. $a \in_R \mathbb{Z}_q, b \in_R \mathbb{Z}_q, u \leftarrow g^a, v \leftarrow g^b$    $\xrightarrow{g,u,v}$
Repeat 2 - 5 $n$ times:
2.                                                           $\xleftarrow{x_i,y_i,z_i}$
3.     if $y_i = {x_i}^a$ then $m_i \leftarrow z_i/{x_i}^b$
4.     else $m_i = \perp$
5.                                                           $\xrightarrow{m_i}$
6.                                                           $\xleftarrow{m'_0,m'_1}$
7. $d \in_R \{0,1\}, r \in_R \mathbb{Z}_q, x \leftarrow g^r$
   $y \in_R G, z' \leftarrow v^r, z \leftarrow m'_d z'$                 $\xrightarrow{x,y,z}$
8.                                                           $\xleftarrow{d'}$
```

Game 15: y is replaced with a random value.

Proof. We transform Game 14 to a new Game 15 by replacing y with a random element.

Then we construct an algorithm D_1 as shown in Algorithm 5 to solve the delayed-target DDH problem. Note D_1 uses A as a subroutine.

```
$D_1^{DDH_{g,g^a},C_{g,g^a}}(g, g^a)$                                    $A$
1. $b \in_R \mathbb{Z}_q, u \leftarrow g^a, v \leftarrow g^b$                         $\xrightarrow{g,u,v}$
Repeat 2 - 5 $n$ times:
2.                                                           $\xleftarrow{x_i,y_i,z_i}$
3.     if $DDH_{g,g^a}(x_i, y_i) = 1$ then $m_i \leftarrow z_i/{x_i}^b$
4.     else $m_i = \perp$
5.                                                           $\xrightarrow{m_i}$
6.                                                           $\xleftarrow{m'_0,m'_1}$
7. $d \in_R \{0,1\}, (x, y) \leftarrow C_{g,g^a}(), z' \leftarrow x^b, z \leftarrow m'_d z'$
8.                                                           $\xrightarrow{x,y,z}$
9.                                                           $\xleftarrow{d'}$
10. if $d' = d$ then return 1
11. else return 0
```

Algorithm 5: D_1

In algorithm D_1, if y is generated by C_{g,g^a} to be $y \leftarrow x^a$, then the computation of A

proceeds as in Game 14, therefore

$$\Pr\left[D_1 = 1 | (y \leftarrow x^a)\right] = \Pr_{14}\left[d' = d\right].$$

If y is generated by C_{g,g^a} to be $y \in_R G$, then the computation of A proceeds as in Game 15, therefore

$$\Pr\left[D_1 = 1 | (y \in_R G)\right] = \Pr_{15}\left[d = d'\right].$$

It follows that

$$\begin{aligned} \left|\Pr_{14}\left[d = d'\right] - \Pr_{15}\left[d = d'\right]\right| &= \left|\Pr\left[D = 1 | (y \leftarrow x^a)\right] - \Pr\left[D = 1 | (y \in_R G)\right]\right| \qquad (2.20) \\ &= Adv_{dtddh}^{D_1}. \end{aligned}$$

Next we transform Game 15 to Game 16 by replacing z' with a random element.

C	A
1. $a \in_R \mathbb{Z}_q, b \in_R \mathbb{Z}_q, u \leftarrow g^a, v \leftarrow g^b$	$\xrightarrow{g, u, v}$
Repeat 2 - 5 n times:	
2.	$\xleftarrow{x_i, y_i, z_i}$
3. if $y_i = {x_i}^a$ then $m_i \leftarrow z_i / {x_i}^b$	
4. else $m_i = \perp$	
5.	$\xrightarrow{m_i}$
6.	$\xleftarrow{m'_0, m'_1}$
7. $d \in_R \{0, 1\}, r \in_R \mathbb{Z}_q, x \leftarrow g^r$ $y \in_R G, z' \in_R G, z \leftarrow m'_d z'$	$\xrightarrow{x, y, z}$
8.	$\xleftarrow{d'}$

Game 16: z' is replaced with a random value.

Then we construct an algorithm D_2 as shown in Algorithm 6 to solve the delayed-target DDH problem.

Let $b = ac$. Then in D_2, we have $v = u^c = g^{ac} = g^b$, and $z' = x^a \Leftrightarrow z'^c = x^{ac} = x^b$. Therefore, if in the challenge pair (x, z'), z' is generated by $z' \leftarrow x^a$, then the computation of A proceeds as in Game 15, and it holds that

$$\Pr\left[D_2 = 1 | (z' \leftarrow x^a)\right] = \Pr_{15}\left[d = d'\right].$$

If in the challenge pair (x, z'), z' is generated by $z' \in_R G$, then the computation of A proceeds as in Game 16, therefore

$$\Pr\left[D_2 = 1 | (z' \in_R G)\right] = \Pr_{16}\left[d = d'\right].$$

```
$D_2^{DDH_{g,g^a},C_{g,g^a}}(g, g^a)$                                   $A$
1. $c \in_R \mathbb{Z}_q, u \leftarrow g^a, v \leftarrow u^c$           $\xrightarrow{g,u,v}$
Repeat 2 - 5 $n$ times:
2.                                                                       $\xleftarrow{x_i,y_i,z_i}$
3.    if $DDH_{g,g^a}(x_i, y_i) = 1$ then $m_i \leftarrow z_i/{y_i}^c$
4.    else $m_i = \perp$
5.                                                                       $\xrightarrow{m_i}$
6.                                                                       $\xleftarrow{m'_0,m'_1}$
7. $d \in_R \{0,1\}, (x, z') \leftarrow C_{g,g^a}(), y \in_R G, z \leftarrow m'_d z'^c$
8.                                                                       $\xrightarrow{x,y,z}$
9.                                                                       $\xleftarrow{d'}$
10. if $d' = d$ then return 1
11. else return 0
```

Algorithm 6: D_2

It follows that

$$\left|\Pr_{15}[d = d'] - \Pr_{16}[d = d']\right| = |\Pr[D_2 = 1|(z' \leftarrow x^a)] - \Pr[D_2 = 1|(z' \in_R G)]| \quad (2.21)$$
$$= Adv_{dtddh}^{D_2}.$$

We also have $\Pr_{16}[d = d'] = 1/2$ since z is independent of m'_b in Game 16. Therefore

$$\left|\Pr_{14}[d' = d] - 1/2\right| \le Adv_{dtddh}^{D_1} + Adv_{dtddh}^{D_2}. \quad (2.22)$$

We conclude that, if DTDDH assumption holds, then DEG is IND-CCA1 secure. □

Decisional DHK1 Assumption

First we review the DHK1 assumption.

Assumption 2.3.9. *The* DHK1 *assumption is as follows: Let G be a group of prime order q, let g be a generator of G, and let $k \approx \log_2 q$. Let $a \in_R \mathbb{Z}_q$. Let O be an oracle that satisfies the following property: when O is queried with a pair $(x_i, y_i) \in G \times G$, O returns r_i such that $x_i = g^{r_i}$ if $y_i = {x_i}^a$. Given (g, g^a), for any polynomial (in k) time algorithm A that has access to O, if A outputs a pair $(x, y) \in G \times G$, then there exists a compiler E such that $A' = E(A)$, and A' satisfies the following conditions: (1) A' is polynomial time; (2) A' has* *the same input, output, oracle access, and random tape access behaviour as A, except that in addition to x and y, A' also outputs r such that*

$$\Pr\left[x = g^r | y = x^a\right] > 1 - \epsilon_{dhk1},$$

or equivalently

$$\Pr\left[x \neq g^r, y = x^a\right] < \epsilon_{dhk1},$$

where ϵ_{dhk1} is negligible.

DHK1 is a generalization of KEA. KEA is the same as DHK1 except that A does not have access to the oracle O. KEA was originally proposed to prove the security of DEG in [30]. Bellare *et al.* pointed out a flaw in the security argument in [30], and proposed DHK1 to prove the security of DEG [9]. The above definition of DHK1 is different from that in [9] in expression, but their ideas are the same.

We observe that using the DDH assumption and the following decisional DHK1 (DDHK1), we can prove that DEG is IND-CCA1 secure.

Assumption 2.3.10. *The* Decisional DHK1 (DDHK1) assumption *is as follows: Let G be a group of prime order q, let g be a generator of G, and let $k \approx \log_2 q$. Let $a \in_R \mathbb{Z}_q$. Let O be an oracle. When O is queried with a pair $(x_i, y_i) \in G \times G$, O returns 1 if $y_i = {x_i}^a$ and returns 0 if $y_i \neq {x_i}^a$. Given (g, g^a), for any polynomial (in k) time algorithm A that has access to O, if A outputs a pair $(x, y) \in G \times G$, then there exists a compiler E such that $A' = E(A)$, and A' satisfies the following conditions: (1) A' is polynomial time; (2) A' has the same input, output, oracle access and random tape access behaviour as A, except that in addition to x and y, A' also outputs a bit b such that*

$$\Pr\left[b = 1, y = x^a\right] + \Pr\left[b = 0, y \neq x^a\right] > 1 - \epsilon$$

where ϵ is negligible.

First, we observe that DHK1 is stronger than DDHK1.

Lemma 2.3.11. *DHK1 implies DDHK1.*

Proof. We consider an assumption DHK1$'$. DHK1$'$ is the same as DHK1 except that the oracle O returns 1 if $y_i = {x_i}^a$ and returns 0 if $y_i \neq {x_i}^a$. We define the event that A wins DHK1 to be the event $g^r \neq x$ and $y = x^a$ in the DHK1 setting, the event that A wins DHK1$'$ to be the event $g^r \neq x$ and $y = x^a$ in the DHK1$'$ setting, and the event that A wins DDHK1 to be the event $b = 0$ and $y = x^a$ in the DDHK1 setting.

First we show that DHK1 implies DHK1$'$. In DHK1$'$, A can tell if $y_i = {x_i}^a$ by querying O. In DHK1, A can not only tell if $y_i = {x_i}^a$, but also get r_i such that $g^{r_i} = x_i$ when $y_i = {x_i}^{r_i}$. It is more likely that A in DHK1 can compute $(x, y = x^a)$ without choosing r where $g^r = x$ than A in DHK1$'$. We use $\Pr_{DHK1'}[e]$ and $\Pr_{DDHK1}[e]$ to denote the probability that an event e happens in DHK1$'$ setting and DDHK1 setting respectively. If A can compute $(x, y = x^a)$ with out knowing $r = \log x$, then A' would not be able to extract r from A. It holds that

$$\begin{aligned}\Pr_{DHK1}[A \text{ wins}] &\geq \Pr_{DHK1'}[A \text{ wins}] \\ \Pr_{DHK1}[g^r \neq x, y = x^a] &\geq \Pr_{DHK1'}[g^r \neq x, y = x^a]\end{aligned}$$

Therefore, if DHK1 holds, then DHK1$'$ holds.

Next we show that DHK1$'$ implies DDHK1. Suppose that DHK1$'$ holds. We construct the algorithm A' in DDHK1 (denoted as A'_{DDHK1}) based on the algorithm A' in DHK1$'$ (denoted as $A'_{DHK1'}$). Let (r, x, y) be the output of the $A'_{DHK1'}$ and let (e, x, y) be the output of A'_{DDHK}. We define that A'_{DDHK} outputs $(1, x, y)$ if $x = g^r$ and $y = (g^a)^r$, otherwise A'_{DDHK} outputs $(0, x, y)$. It holds that

$$\Pr[e = 0 | y \neq x^a] = 1$$

and

$$\begin{aligned}\Pr[e = 1 | y = x^a] &\geq \Pr[e = 1, x = g^r | y = x^a] \\ &= \Pr[e = 1 | x = g^r, y = x^a] \Pr[x = g^r | y = x^a] \\ &> 1 - \epsilon_{dhk1}.\end{aligned}$$

It holds that

$$\begin{aligned}& \Pr[e = 1, y = x^a] + \Pr[e = 0, y \neq x^a] \\ = \ & \Pr[e = 1 | y = x^a] \Pr[y = x^a] + \Pr[e = 0 | y \neq x^a] \Pr[y \neq x^a] \\ > \ & (1 - \epsilon_{dhk1}) \Pr[y = x^a] + \Pr[y \neq x^a] \\ = \ & 1 - \epsilon_{dhk1} \Pr[y = x^a].\end{aligned}$$

Therefore, if DHK1$'$ holds, then DDHK1 holds.

We conclude that DHK1 implies DDHK1. □

Next, we prove that the DDH and DDHK1 assumptions imply the DTDDH assumption.

Theorem 2.3.12. *DDH and DDHK1 assumptions imply DTDDH assumption.*

C	A
1. $a \in_R \mathbb{Z}_q, u = g^a$	$\xrightarrow{g, u}$
Repeat 2 - 3 n times:	
2.	$\xleftarrow{x_i, y_i}$
3. If $y_i = {x_i}^a$ then $e'_i \leftarrow 1$, otherwise, $e'_i \leftarrow 0$	$\xrightarrow{e'_i}$
4.	$\xrightarrow{x, y}$
5.	$\xleftarrow{e}$
6. return e	

Game 17: DTDDH game

Proof. we define a DTDDH game as shown in Game 17. Let Adv^A_{DTDDH} be the probability that A answers e correctly in a game.

We transform Game 17 to Game 18 by replacing A with $A' = E(A)$ where E and A' are as defined in the DDHK1 assumption.

C	A'
1. $a \in_R \mathbb{Z}_q, u = g^a$	$\xrightarrow{g, u}$
Repeat 2 - 3 n times:	
2.	$\xleftarrow{e_i, x_i, y_i}$
3. If $y_i = {x_i}^a$ then $e'_i \leftarrow 1$, otherwise, $e'_i \leftarrow 0$	$\xrightarrow{e'_i}$
4.	$\xrightarrow{x, y}$
5.	$\xleftarrow{e}$
6. return e	

Game 18:

Let $Adv^{A'}_{game_{18}}$ be the probability that A' answers e correctly in a game. It holds that

$$Adv^A_{dtddh} = Adv^{A'}_{game_{18}}.$$

Next we transform Game 18 to 19 in which C set $e'_i \leftarrow e_i$.

If the DDHK1 assumption holds, then in Step 3, e'_i is a correct answer with probability $(1 - \epsilon)$ where ϵ is negligible. If in all n rounds, all the e'_i values are correct, then the computation of A' proceeds the same way as in Game 18. Therefore, it holds that

$$Adv^{A'}_{game_{19}} \geq Adv^{A'}_{game_{18}}(1 - \epsilon)^n$$

C		A'
1. $a \in_R \mathbb{Z}_q, u = g^a$	$\xrightarrow{g, u}$	
Repeat 2 - 3 n times:		
2.	$\xleftarrow{e_i, x_i, y_i}$	
3. $e'_i \leftarrow e_i$	$\xrightarrow{e'_i}$	
4.	$\xrightarrow{x, y}$	
5.	$\xleftarrow{e}$	
6. return e		

Game 19:

Note that in Game 19, C does not use a to compute e'_i. We can transform Game 19 to Game 20 where C takes a triple $(g^a, x, y) \in G^3$ as input and solves the DDH problem.

$C(g^a, x, y)$		A'
1. $u = g^a$	$\xrightarrow{g, u}$	
Repeat 2 - 3 n times:		
2.	$\xleftarrow{e_i, x_i, y_i}$	
3. $e'_i = e_i$	$\xrightarrow{e'_i}$	
4.	$\xrightarrow{x, y}$	
5.	$\xleftarrow{e}$	
6. return e		

Game 20:

Let Adv^C_{DDH} be the probability that C correctly answers if $y = x^a$. It holds that

$$Adv^C_{DDH} = Adv^{A'}_{game_{19}}.$$

It follows that

$$Adv^C_{ddh} \geq Adv^A_{dtddh}(1 - \epsilon)^n.$$

Note that n is polynomial in k and ϵ is negligible in k if the DDHK1 assumption holds. We conclude that if the DDDHK1 assumption holds, then the DDH assumption implies the DTDDH assumption; i.e., the DDH assumption and the DDHK1 assumption imply the DTDDH assumption. $\square$

The above results prove that DHK1 is stronger than necessary in the security proof for DEG.

Chapter 3

Sensor Networks Key Management

In this chapter, we first study connectivity of key predistribution schemes in Section 3.1. In Section 3.1.1, we study the connectivity of the block graphs of a class of deterministic KPS based on transversal designs. In Section 3.1.2, we study the κ-edge-connectivity of a random KPS. In Section 3.1.3, we study the κ-edge-connectivity of two classes of DSNs, one based on the deterministic KPS derived from transversal designs, and the other based on random KPS.

In Section 3.2, we study multi-path key establishment. In Section 3.2.1, we describe the proposed model and some preliminary results on secure message transmission, Reed-Solomon codes and key derivation using resilient functions. In Section 3.2.3, we present and analyze the HM and JERT schemes. In Section 3.2.4, we present our first two new schemes and their analysis. In Section 3.2.5, we present our third scheme, which tolerates less than 1/2 of the paths controlled by the adversary.

3.1 κ-connectivity of Key Predistribution

3.1.1 Transversal Design KPS

κ-connectivity of the Block Graph

In [70], [71], Lee and Stinson proposed a deterministic KPS based on transversal design TD(k, r) [101, §6.6]. The TD(k, r) KPS has the following properties:

- The scheme uses kr keys, which are divided into k sets of r keys.
- Each node is assigned k keys, one from each set.
- Every pair of keys from different sets is contained in exactly one node.
- Each key is assigned to exactly r nodes.
- Each node has degree equal to $k(r-1)$ in the block graph G_B.
- The number of nodes is r^2.

The designs TD(k, r) exist for a wide variety of parameters k and r. For example, TD(k, r) are known to exist for all pairs (k, r) where r is a prime or prime power and $k \leq r$.

In this section, we prove that the block graph G_B of a TD(k, r) KPS is $k(r-1)$-edge-connected. (In fact, the connectivity cannot be greater than this value because G_B is a regular graph of degree $k(r-1)$.) The proof idea is from [53], in which the connectivity of two combinatorial structures, BIBD (balanced incomplete block design) and PBD (pairwise balanced design), are proved.

Theorem 3.1.1. *The block graph of a TD(k, r) KPS is $k(r-1)$-vertex-connected and $k(r-1)$-edge-connected.*

Proof. First we consider the vertex connectivity. Let B the set of vertices in G_B.

Let a and c be any two nonadjacent vertices in G_B. Let C be a vertex cut; i.e., a set of nodes, the removal of which separates a and c.

Let x be a key not assigned to a or c (i.e., $x \notin a \cup c$). Let $S_a(x)$ be the nodes containing x that are adjacent to a in G_B, and let $S_c(x)$ be the nodes containing x that are adjacent to c in G_B. Suppose there exist nodes b_1 and b_2 such that $b_1 \in S_a(x)$, $b_1 \notin C$, $b_2 \in S_c(x)$ and $b_2 \notin C$. Then $a\ b_1\ b_2\ c$ is a path in $G_B - C$ since b_1 and b_2 are adjacent (they both contain the key x). This contradicts the assumption that C separates a and c. From this, it follows immediately that either $S_a(x) \backslash S_c(x) \subseteq C$ or $S_c(x) \backslash S_a(x) \subseteq C$.

For $x \notin a \cup c$, there are exactly $k-1$ pairs of keys which contain x and a key in a from one of the $k-1$ columns which do not contain x. Each of these $k-1$ pairs of keys is contained in a different node. So there are $k-1$ nodes which contain x and are adjacent to a in G_B. Similarly there are $k-1$ nodes which contain x and are adjacent to c in G_B. Then

$$\begin{aligned}
|S_a(x) \backslash S_c(x)| &= |S_a(x)| - |S_a(x) \cap S_c(x)| \\
&= k - 1 - |S_a(x) \cap S_c(x)| \\
&= |S_c(x)| - |S_a(x) \cap S_c(x)| \\
&= |S_c(x) \backslash S_a(x)|.
\end{aligned}$$

Let B' be the nodes in $B \backslash \{a, c\}$ which are adjacent to exactly one of a or c in G_B, and let $C' = B' \cap C$. From the above calculation, it follows that at least half of the nodes that are in B' and contain x are in C'.

Each node $b \in C'$ has exactly one key shared with a or c. Therefore,

$$\begin{aligned}
2|C'|(k-1) &= 2 \sum_{b \in C'} |\{(x, b) : x \notin a \cup c, x \in b\}| \\
&= 2 \sum_{x \notin a \cup c} |\{(x, b) : b \in C', x \in b\}| \\
&\geq \sum_{x \notin a \cup c} |\{(x, b) : b \in B', x \in b\}| \\
&= \sum_{b \in B'} |\{(x, b) : x \notin a \cup c, x \in b\}| \\
&= |B'|(k-1).
\end{aligned}$$

Hence, $|C'| \geq |B'|/2$.

Next, let B'' be the nodes adjacent to both a and c. There are $k(k-1)$ pairs of keys such that one key in the pair is in a and the other is in c. These pairs are all contained in different nodes. So $|B''| = k(k-1)$.

Each one of the k keys in a occurs in $r-1$ other nodes. Of these $r-1$ nodes, there are $k-1$ nodes adjacent to c. So there are $k(r-k)$ nodes that are adjacent to a but not adjacent to c. Similarly, there are $k(r-k)$ nodes that are adjacent to c but not adjacent to a. Therefore we have $|B'| = 2k(r-k)$.

Finally, we have that

$$|C| \geq |C'| + |B''| \geq \frac{|B'|}{2} + |B''| = k(r-1).$$

Therefore, G_B is at least $k(r-1)$-vertex-connected. Also, it is at least $k(r-1)$-edge-connected (by Whitney's inequality [109]). Since each node has $k(r-1)$ edges, we conclude that G_B is exactly $k(r-1)$-vertex-connected and $k(r-1)$-edge-connected. $\square$

3.1.2 Random KPS

Minimum Degree of the Block Graph

In a random KPS, for each of the n nodes, k keys are randomly drawn from a key pool of size v ([40]). The block graph is called a *random intersection graph* and denoted $G(n, v, k)$.

In some previous papers (e.g., [40], [26], [37], [108], [59]), the random intersection graph $G(n, v, k)$ is implicitly assumed to be a random graph $G(n, p_1)$, where two nodes are adjacent with probability

$$p_1 = 1 - \frac{\binom{v-k}{k}}{\binom{v}{k}}.$$

Note that p_1 is the probability that two nodes share at least one key. For clarity, we now make this assumption explicit:

Assumption 3.1.2. *A random intersection graph $G(n, v, k)$ can be modelled as a random graph $G(n, p_1)$ where $p_1 = 1 - \frac{\binom{v-k}{k}}{\binom{v}{k}}$.*

Given the above assumption, we would compute the κ-edge-connectivity of the $G(n, p_1)$, and then expect that the $G(n, v, k)$ is κ-edge-connected as well. The minimum degree of a graph is a bound on its connectivity. Therefore we study the minimum vertex degree in $G(n, p_1)$.

The degrees of the vertices in $G(n, p_1)$ are random variables which follow binomial distributions with parameters $n - 1$ and p_1. These random variables can be considered as independent [18, §3]. We can obtain the expected minimum degree using minimum order statistics [31]. We have the following result:

Theorem 3.1.3. *The expected minimum value among n independent random variables is*

$$\overline{d_{min}} = \sum_{i=0}^{n-1} (1 - F(i))^n - n(1 - F(n))^n \tag{3.1}$$

where $F(x)$ is the cumulative probability distribution of the random variables.

Proof. Let $F_{min}(x) = \Pr[d_{min} \leq x]$ be the cumulative distribution of the minimum value d_{min} of the n random variables $d_1, \cdots, d_n$. Then we have

$$\begin{aligned} F_{min}(x) &= \Pr[d_{min} \leq x] \\ &= 1 - \Pr[d_1 > x \text{ and } \cdots \text{ and } d_n > x] \\ &= 1 - (1 - F(x))^n. \end{aligned}$$

Let $f_{min} = \Pr[d_{min} = x]$ be the mass distribution of d_{min}. Then

$$f_{min}(x) = F_{min}(x) - F_{min}(x-1). \tag{3.2}$$

Let $\overline{d_{min}}$ be the expectation of d_{min}. We have

$$\begin{aligned}
\overline{d_{min}} &= \sum_{x=0}^{n} x f_{min}(x) \\
&= \sum_{x=1}^{n} x(F_{min}(x) - F_{min}(x-1)) \\
&= nF_{min}(n) - \sum_{x=0}^{n-1} F_{min}(x) \\
&= n(1-(1-F(n))^n) - \sum_{x=0}^{n-1}(1-(1-F(x))^n) \\
&= \sum_{x=0}^{n-1}(1-F(x))^n - n(1-F(n))^n
\end{aligned}$$

$\square$

We know that for $G(n, p_1)$, it holds that

$$\begin{aligned}
F(x) &= \Pr[\mathrm{d} \le x] \\
&= \sum_{i=0}^{x} \binom{n-1}{i} {p_1}^i (1-p_1)^{n-1-i}.
\end{aligned} \tag{3.3}$$

Using (3.1) and (3.3), we can compute the expected minimum degree of $G(n, p_1)$, and use it to estimate the expected minimum degree of $G(n, v, k)$. We use simulation to verify the result. In the simulation, we set $n = 2000, k = 5, v = 500, 1000, 1500, 2000, 2500, 3000$, and 3500. We randomly choose k out of v keys for each node, construct a graph by creating edges between nodes who share keys, and compute the minimum degree in the resulting graphs. For each v we run the simulation 100 times. The average minimum degree in the simulation and the estimated minimum degree computed using the formulas developed above are listed in Table 3.1. From this table, it can be seen that the estimation is very close to the simulation.

We also use simulation to verify $f_{min}(x)$, the distribution of d_{min} computed using (3.2). We choose $n = 2000, k = 5$, and $v = 2000$, and run the simulation 200 times. The results are shown in Table 3.2. We find that computed values match the simulation well.

v	$\overline{d_{min}}$ in simulation	Computed $\overline{d_{min}}$
500	67.36	66.86
1000	27.61	27.53
1500	15.27	15.43
2000	9.80	9.80
2500	6.48	6.65
3000	4.62	4.68
3500	3.31	3.37

Table 3.1: Expected minimum degree of a block graph for a random KPS.

κ-connectivity of the Block Graph of a Random KPS

Since the random graph $G(n, p_1)$ is $\overline{d_{min}}$-edge-connected asymptotically [18], by Assumption 3.1.2, we estimate $G(n, v, k)$ to be $\overline{d_{min}}$-edge-connected as well. We verify this assumption using simulations. We use Maple 9.5 to compute the minimum degree and edge-connectivity of a given graph. Because finding the connectivity of a graph is time consuming, we only test $G(n, v, k)$ with small number of nodes. In the simulation, we set $n = 100$, $k = 5$, $v = 100, 200, 300$, and check the minimum degree and minimum edge cut of the resulting $G(n, v, k)$. We run the simulation 10 times with different random seeds for each v value. In all the 30 simulations, the minimum degree and the connectivity of $G(n, v, k)$ are the same (the range of values is from 1 to 14).

3.1.3 κ-connectivity of the DSN

After the sensor nodes are deployed, only the pairs of nodes located within each other's wireless communication range can communicate directly. In the previous KPS papers ([40], [26], [37], [108], [59]), the random geometric graph G_G is implicitly assumed to be a random graph. Observe that the network graph G_N can be viewed as being generated from the block graph G_B by deleting some edges, namely, those which are not in the graph G_G. An edge in G_B will thus be deleted with probability $1 - p_2$, where p_2 is the probability that two nodes are in each other's wireless communication range. The parameter p_2 is determined by the the area in which the nodes are deployed, the number of nodes deployed, and by the nodes' wireless communication range.

Here we make the assumption regarding the random geometric graph G_G explicit:

x	$f_{min}(x)$ in simulation	Computed $f_{min}(x)$
3	0	0
4	0	0
5	0	0
6	0.01	0.01
7	0.03	0.03
8	0.09	0.1
9	0.23	0.22
10	0.35	0.33
11	0.26	0.25
12	0.04	0.05
13	0	0
14	0	0
15	0	0

Table 3.2: Distribution of minimum degree of random KPS block graph.

Assumption 3.1.4. *We assume the random geometric graph G_G with parameters n and p_2 can be modelled as a random graph $G(n, p_2)$; i.e., we assume that G_G is generated from a complete graph by deleting every edge independently, with probability $1 - p_2$.*

Based on this assumption, we can estimate the expected minimum node degree of the network graph G_N. We will again verify the result by simulation.

Minimum Degree of DSN Constructed from a TD-based KPS

The block graph G_B for a KPS constructed from a transversal design TD(k, n) is a regular graph with degree $k(r-1)$. After deleting each edge with probability $1-p_2$, the degree of a given node is a random variable following a binomial distribution with parameters $k(r-1)$ and p_2, and the random variable's cumulative probability distribution is as follows:

$$\begin{aligned} F(x) &= \Pr[d \le x] \\ &= \sum_{i=0}^{x} \binom{k(r-1)}{i} {p_2}^i (1-p_2)^{k(r-1)-i}. \end{aligned} \qquad (3.4)$$

In our simulation, we first assign keys to the nodes using a TD(k, n), and then we distribute the nodes uniformly at random in an $L \times L$ square. We delete an edge between two nodes in the block graph if the Euclidean distance between the two nodes is greater than ℓ, where ℓ is the wireless communication range of a node. To eliminate the "border effect" of the geometric distribution, we do not use the nodes close to the border when computing the minimum degree[1]. In the simulation, we set $k = 30, r = 49, v = 1470, n = 2401$ and $L = 100$. We compute $p_2 = \frac{\pi l^2}{L^2}$, we use (3.1) and (3.4) to compute $\overline{d_{min}}$, and we compare the results with simulation in Table 3.3. We find that the results match well.

l	$\overline{d_{min}}$ in simulation	Computed $\overline{d_{min}}$
4	0.04	0.18
5	1.32	1.79
6	4.39	4.37
7	7.61	7.90
8	12.06	12.36
9	17.79	17.74
10	23.93	24.05

Table 3.3: Minimum degree of a DSN using a TD KPS

Minimum Degree of DSN Constructed from a Random KPS

In section 3.1.2, we assumed the block graph G_B of a random KPS is a random graph where an edge joins each pair of nodes with probability p_1. After the nodes are deployed, each existing edge will be deleted with probability $1 - p_2$. Under Assumptions 3.1.2 and 3.1.4, the resulting network can be considered as a random graph in which each edge is included with probability $p_1 p_2$. The probability distribution of an edge degree follows a binomial distribution with parameters $n - 1$ and $p_1 p_2$, and its cumulative probability distribution is

$$\begin{aligned} F(x) &= \Pr[\mathrm{d} \leq x] \\ &= \sum_{i=0}^{x} \binom{n-1}{i} (p_1 p_2)^i (1 - p_1 p_2)^{n-1-i}. \end{aligned} \tag{3.5}$$

We use (3.1) and (3.5) to compute the expected minimum degree $\overline{d_{min}}$, and compare it with a value obtained by simulation. The simulation setting is the same as in Section 3.1.3 except that we assign keys using the random KPS. The results are shown in Table 3.4 and they match well.

[1]For a detailed discussion on border effect and methods to avoid border effect, see [13].

l	$\overline{d_{min}}$ in simulation	Computed $\overline{d_{min}}$
4	0.00	0.00
5	0.38	0.72
6	2.36	2.41
7	4.58	4.86
8	7.64	8.03
9	11.83	11.92
10	16.52	16.52

Table 3.4: Minimum degree of a DSN constructed from a random KPS

κ-connectivity of DSN

Based on Assumption 3.1.2 and Assumption 3.1.4, the network graph G_N of the DSN constructed from a random KPS is also a random graph. A random graph is asymptotically κ-edge-connected when its minimum degree is κ [88]. Thus G_N can be estimated to be $\overline{d_{min}}$-edge-connected on average, where $\overline{d_{min}}$ is computed using (3.1) and (3.5).

For the DSN constructed from a TD-type KPS, we make the following assumption:

Assumption 3.1.5. *The network graph G_N generated by intersecting the block graph of a TD-type KPS and a random geometric graph is κ-edge-connected, where its minimum degree is κ.*

The assumption is based on the observation that in the random graph model and the random geometric graph model, when the graph's minimum node degree is κ, the graph becomes κ-edge-connected asymptotically ([18], [88]). Under this assumption, we can estimate the DSN constructed from a TD-type KPS to be $\overline{d_{min}}$-connected, where $\overline{d_{min}}$ can be computed using (3.1) and (3.4).

We verify the above estimate with simulations using Maple 9.5. In the simulation, we compare the minimum degree and κ-edge-connectivity of G_N. Because finding the κ-edge-connectivity of a graph is time-consuming, we only test network graphs G_N with small number of nodes. We use $n = 121, v = 55, k = 5$ for the key assignment schemes. For geometric distribution we distribute the nodes in a 100×100 grid as in section 3.1.3. We set $\ell = 30, 40, 50$ respectively. For each key distribution schemes and each ℓ-value, we run the simulation 10 times. In all 60 simulations (30 for the DSNs constructed from a random KPS and 30 for the DSNs constructed from a TD-type KPS), the minimum degree and the size of the minimum cut of G_N are identical(these values range from 0 to 12).

Note that, since the $\overline{d_{min}}$ value computed using (3.5) or (3.4) reflects the minimum degree of the nodes not on the border of the network's geometric area, it reflects the connectivity of the "majority" of the network, excluding the nodes on the border of the geometric area.

3.2 Multi-path Key Establishment

3.2.1 Model and Preliminaries

Our model for multi-path key establishment (MPKE) is an enhancement of the model used in [57] and [32]. It is described as follows.

1. In a sensor network secured using key predistribution schemes, there often are multiple node-disjoint paths between a specified source node A and a specified destination node B. Every two consecutive nodes (i.e., a *link*) on such a path have a common key, and no two of these paths contain any common nodes except for A and B. These paths are identified by A before key establishment takes place.[2] A sends key establishment messages over the paths. The efficiency of the scheme is measured by communication complexity; i.e., the total amount of information that is transmitted over all the paths.

2. We will assume that a fraction e of these paths (where $0 \leq e < 1/2$) are controlled by an adversary. We call e the *error rate*. A path P is *controlled by an adversary* if there exists an adversary that has knowledge of the key corresponding to a link on the path P. We assume that an adversary controlling a path P can observe, drop or alter any messages that are transmitted from A to B using the path P.

3. The goal of a key establishment scheme is to enable A and B to establish a key with sufficient entropy. This leads to two security requirements. First, the adversary should not be able to disrupt the protocol by preventing B from computing the same key K that A holds. Second, the adversary should be prevented from determining partial information about the established key. This idea is formalized by considering the entropy of the message M received by B, from the point of view of the adversary who collects partial information about M (see Section 3.2.2 for details).

   Entropy of the established key is not analyzed in [57] and [32]. However, entropy is a critical requirement for the established key to be secure, because a key with low

[2]In the previous model in [57] and [32], it is assumed that some routing protocols are used to identify the paths. In this model, we follow the assumption.

entropy can easily be determined by the adversary by exhaustive search. Key entropy should be considered when evaluating the scheme, along with efficiency.

Remark. An adversary who controls a node N in one of the paths P from A to B has access to all the keys stored in N. Suppose that one of these keys, say K, is stored by a node N' in another path P' from A to B. Therefore the adversary can read information encrypted using the key K, and thus the path P' will not be secure. So the number of paths controlled by the adversary can be greater than the number of nodes controlled by the adversary. See [107] for more discussion on this issue.

Secure Message Transmission

Perfectly secure message transmission (PSMT) was introduced in 1993 by Dolev *et al.* [36]. PSMT was first suggested for use in multi-path key establishment in sensor networks in 2004 by Wang [107]. We define PSMT protocols and summarize some relevant results in this section.

Suppose two parties A and B are connected by p channels. An adversary controls p_a (or fewer) of these channels, but it is not known which channels are controlled by the adversary. The adversary can observe, delete, or modify the information in these p_a channels. An *r-round (p, p_a)-perfectly secure message transmission scheme* is an interactive protocol between A and B which takes place in r rounds (denoted as rounds $1, \dots, r$), such that the following properties are satisfied:

1. In each odd-numbered round, A sends information to B over each of the p channels connecting them.

2. In each even-numbered round, B sends information to A over each of the p channels connecting them.

3. After the rth round, A and B both possess a common key K which is an element of a prespecified key space $\mathcal{K}$.

4. The adversary has no information on the value of K (so the entropy of K, from the point of view of the adversary, is $\log|\mathcal{K}|$).

The *overhead* of a PSMT is defined to be the ratio

$$\frac{\text{amount of information transmitted over all } p \text{ channels}}{\text{length of the key } K}.$$

There is a large literature on PSMT. For our purposes, we are most interested in one-round protocols, since these are the simplest and best suited to be applied to multi-path key establishment. It was proven in [36] that a 1-round (p, p_a)-perfectly secure message transmission scheme exists if and only if $p \geq 3p_a + 1$. It is shown in [45] that the overhead of a one-round PSMT satisfies the condition

$$\text{overhead} \geq \frac{p}{p - 3p_a}. \tag{3.6}$$

Furthermore, for all pairs (p, p_a) with $p \geq 3p_a + 1$, schemes that meet the bound in (3.6) with equality (i.e., *optimal overhead schemes*) are constructed in [45].

If $3p_a \geq p \geq 2p_a + 1$, then all is not lost. It is possible to construct 2-round (p, p_a)-PSMT in these cases [91, 67]. Alternatively, one can obtain one-round schemes that are not perfectly reliable (i.e., condition 3 in the definition of PSMT is relaxed). Such schemes are constructed in [66, 87].

Comparison Between PSMT and MPKE

A PSMT assumes multiple channels connecting A and B. The channels not controlled by the adversary are assumed to provide unconditional secrecy and authenticity. The security of a PSMT scheme is unconditional, provided that no computational assumptions are made in the analysis of the protocol. Almost all PSMT in the literature are studied in the setting of unconditional security.

In an MPKE, information is transmitted over links in encrypted form using conventional secret-key cryptography. Therefore we do not expect an MPKE to provide unconditional security; the security will depend on the assumption that the encryption and authentication schemes are (computationally) secure. Additional computational assumptions may be required, depending on the scheme.

3.2.2 Reed-Solomon Codes

Many PSMT protocols are based on Reed-Solomon codes, which we introduce now. There are different ways to construct RS codes. Each has its encoding and decoding algorithms. Here we only describe the general functionalities of the encoding/decoding algorithms. For details of the algorithms, see, e.g., [89, 58]. Simply speaking, the input of the RS encoding algorithm is a *message* $\mathbf{m} = (m_0, \ldots, m_{k-1}) \in F_q^k$ where F_q is a finite field of order q. The output of an RS encoding algorithm is $\mathbf{c} = (c_0, \ldots, c_{n-1}) \in F_q^n$ where $k \leq n \leq q$. $\mathbf{c}$ is called a *codeword*. Each coordinate in $\mathbf{m}$ or $\mathbf{c}$ is called a *symbol*.

If it always happens that $c_i = m_i$ for $0 \leq i \leq k-1$, then the encoding is *systematic*. In this case, $m_0, \ldots, m_{k-1}$ may be called *information symbols* and $c_k, \ldots, c_{n-1}$ may be called *parity check symbols*. Not all RS encoding schemes are systematic, however.

The above-described RS code has *length* n and *dimension* k. Its *distance* is $d = n - k + 1$ (i.e., any two distinct codewords differ in at least $n - k + 1$ symbols).

A Reed-Solomon code is a *linear code*, which means that the codewords form a k-dimensional subspace of the vector space F_q^n. A commonly-used method of encoding a linear code is to constuct a *generator matrix*, denoted G, whose rows form a basis for the code. Then, to encode a message $\mathbf{m}$, we compute $\mathbf{c} = \mathbf{m}G$.

During transmission, some symbols in a codeword $\mathbf{c}$ may be deleted or altered. Suppose that δ of the symbols in $\mathbf{c}$ are deleted, and ϵ other symbols in $\mathbf{c}$ are altered. Let $\mathbf{r}$ be the resulting *received vector*. The input of the decoding algorithm is $\mathbf{r}$. The output of the decoding algorithm is the codeword whose distance from $\mathbf{r}$ is minimized. It is a standard result in coding theory that this decoding algorithm will output $\mathbf{c}$ provided that

$$\delta + 2\epsilon < d. \tag{3.7}$$

In the case of an RS code, we have $d = n - k + 1$ and the condition (3.7) becomes

$$\delta + 2\epsilon \leq n - k. \tag{3.8}$$

Given any codeword $\mathbf{c}$, it is a simple matter to obtain the corresponding message $\mathbf{m}$, regardless of whether or not the code is systematic.

In the above-described RS code, a message consists of k symbols and a codeword consists of n ($n \geq k$) symbols. The code is termed an (n, k) RS code.

Evaluating the Secrecy of a Message

When RS codes are used to encode and transmit a secret message $\mathbf{m} = (m_0, \ldots, m_{k-1})$, we need to consider the *entropy* of $\mathbf{m}$ from an adversary's point of view. The original entropy of $\mathbf{m}$ is $k \log_2 q$ bits. Suppose that i symbols are received by the adversary. If $i \geq k$, then the adversary can recover $\mathbf{m}$, and the entropy of $\mathbf{m}$ is 0. If $i < k$, then the adversary can randomly choose $k - i$ other symbols and recover a (possibly incorrect) message. In this case the adversary recovers the correct message with probability

$$\frac{1}{2^{(k-i)\log_2 q}}.$$

Therefore, when the adversary knows i symbols in $\mathbf{m}$, the entropy of $\mathbf{m}$ is

$$\max\{(k-i)\log_2 q, 0\}$$

bits.

Key Derivation and Resilient Functions

In the protocols we will be describing, the key K, which is derived from a k-tuple $\mathbf{m}$, should have sufficient entropy. We will use the number of symbols, instead of number of bits, to indicate the entropy. In this terminology, the entropy of a message $\mathbf{m}$ is k symbols.

To ensure that K is secure, we desire that $\mathbf{m}$ should have entropy at least ℓ symbols, for some prespecified value of ℓ. There are several ways to derive K from $\mathbf{m}$ while preserving its entropy. For example, we can use a cryptographic hash function *hash* to compute $K = \mathit{hash}(\mathbf{m})$. If it holds that

1. the hash function is modelled as a random oracle,
2. the input of the hash function has entropy at least ℓ symbols, and
3. the output of the hash function has a length of at least ℓ symbols,

then the entropy of the derived key K is at least ℓ symbols (so we say that K is *ℓ-secure*).

The above approach only provides computational security of the key. An alternative is to use resilient functions [11, 27] to derive the key. This approach would provide unconditional security of the key.

Suppose q is a prime power. Let k, ℓ, t be positive integers such that $k \geq \ell + t$. A *(k, ℓ, t, q)-resilient function*, or *(k, ℓ, t, q)-RF*, is a function $f : F_q^k \rightarrow F_q^\ell$ such that $f(\mathbf{m})$ is uniformly distributed in F_q^ℓ whenever any t inputs are fixed and the remaining $k - t$ inputs are chosen independently and uniformly at random from F_q, e.g., by an adversary (here we are regarding f as a function with k inputs from F_q).

There is a large body of literature on resilient functions. For our purposes, we need a well-known class of resilient functions that is derived from Reed-Solomon codes. In fact, any linear code gives rise to a linear function. The following was proven for binary codes in [11, 27]. It was observed in [99] that the same result holds for codes over an arbitrary finite field.

Theorem 3.2.1. *Suppose q is a prime power, and suppose there exists a linear code over F_q having length n, dimension k and distance d. Then there exists a $(n, k, d - 1, q)$-RF.*

Using Reed-Solomon codes, the following is an immediate corollary.

Corollary 3.2.2. *Suppose q is a prime power such that $q \geq k > \ell$, where k and ℓ are positive integers. Then there exists a $(k, \ell, k - \ell, q)$-RF.*

The construction of a $(k, \ell, k - \ell, q)$-RF is easy. Let G be the generator matrix of a Reed-Solomon code of dimension ℓ and length k over F_q. Then f is defined as $f(\mathbf{m}) = \mathbf{m}G^T$, where G^T denotes the transpose of G.

Remark. The above-described usage of resilient functions has previously been employed in the literature on PSMT; see, for example, the function **EXTRAND** in [87, §4.2]. However, the connection to resilient functions is not made in [87] or in other papers on PSMT.

3.2.3 Analysis of the HM and JERT Schemes

The HM Scheme

The HM scheme [57] is as follows. Let $n - k = 2t$. The number of paths controlled by the adversary is assumed to be at most t. Suppose that there are p node-disjoint paths between A and B, where $2t < p \leq k$. A chooses a message $\mathbf{m} = (m_0, \dots, m_{k-1})$ and uses a systematic (n, k) RS encoding algorithm to generate a codeword $\mathbf{c} = (m_0, \dots, m_{k-1}, b_0, \dots, b_{2t-1})$. $m_0, \dots, m_{k-1}$ are k information symbols and $b_0, \dots, b_{2t-1}$ are $2t$ parity check symbols. Let $\mathbf{b} = (b_0, \dots, b_{2t-1})$. Then A creates k $(2t + 1)$-tuples, each of the form $m_i \parallel \mathbf{b}$, and sends at most t of the $(2t + 1)$-tuples on each of the p node-disjoint paths (note that this requires that $k \leq pt$, which is not stated as a necessary condition in [57]).

Since there are at most t paths that are controlled by the adversary and $p > 2t$, B can use the majority rule to find the correct $\mathbf{b}$. It is then claimed in [57] that B can then recover $\mathbf{m}$, but the ability to recover $\mathbf{m}$ also depends on how many information symbols have been altered by the adversary. In fact, we show that B may not be able to recover $\mathbf{m}$ at all in many situations. Suppose $k > p > 2t$ (note that it is assumed that $2t < p \leq k$, so we are just saying that $k \neq p$). Suppose that each message symbol is transmitted by one path. Then there is at least one of the p paths, say P_0, that is used to transmit at least two message symbols. If P_0 is one of the t paths controlled by the adversary, then the adversary can alter at least $t + 1$ message symbols. However, an RS code can only correct t errors, so the message cannot be recovered by B.

Another problem with the scheme in [57] is that adversary can obtain information about the message. Recall that the scheme is supposed to tolerate up to t compromised paths. However, if t paths are controlled by the adversary, then the adversary receives $2t$ correct parity check symbols and at least t correct information symbols, and hence by (3.8) the adversaries collectively are able to recover $\mathbf{m}$ when $3t \geq k$ (equivalently, when $n \leq 5t$). Even when there is only one path controlled by the adversary, it will receive $2t$ parity check symbols and at least one information symbol. Then the entropy of the key is $\max\{k-2t-1, 0\}$

symbols, which could be very low.

We regard it as a weakness in the scheme for A to send all the $2t$ parity check symbols on every path, because for decoding of RS codes, a parity check symbol yields the same amount of information about the message as an information symbol does. Another problem (as noted in [32]) is that the scheme is quite inefficient due to the amount of repeated information that is transmitted.

The JERT Scheme

JERT [32] is designed for two neighbouring nodes that have direct but insecure communication. In this case, A and B can run a challenge-response authentication protocol over this insecure channel to verify if they share a common secret key. The communication overhead over the direct link may be neglected.

Here are the details of the scheme. A chooses $\mathbf{m} = (m_0, \ldots, m_{k-1})$, encodes it into a codeword $\mathbf{c} = (c_0, \ldots, c_{n-1})$, and derives a key K from $\mathbf{m}$. Then A selects p node-disjoint paths between A and B. A divides the n symbols into R groups. Group j contains r_j symbols. It holds that

$$\sum_{j=1}^{R} r_j = n.$$

A sends the n symbols in R rounds. In round j, the r_j symbols in group j are sent over the p paths. For each path i, A computes *fraction parameters* q_i $(0 \leq q_i \leq 1)$, where

$$\sum_{i=1}^{p} q_i = 1.$$

Then A sends $r_j q_i$ symbols over path i in round j. In each round, if B can recover an $\mathbf{m}'$ using all the received symbols, then B derives a key K' from $\mathbf{m}'$, and runs an authentication protocol with A over their direct link to verify if $K = K'$. A keeps on sending the codeword symbols until the authentication protocol indicates that $K' = K$ or until all n symbols are sent.

The main purpose JERT is for A to send *just enough* symbols for B to recover K, instead of transmitting all symbols as in the HM scheme. JERT may be thought of as an *adaptive* algorithm, whereas HM is *non-adaptive.*

The security analysis in [32] discusses three attack scenarios:

1. All adversary nodes are passive. In this case, the probability that a given fraction of symbols is received by the adversary nodes is computed.

2. All adversary nodes are active. In this case, the number of symbols that must be sent so that B can recover the key is computed.

3. Some adversary nodes are active and some are passive. This case is not analyzed in [32], where it is stated that an analysis of this case "would be quite complex".

In our schemes, we consider all possible attack scenarios. First, the adversary cannot make B accept an incorrect key, even if all the adversary nodes are active. Since B learns the correct key, K, we only need to consider how much information the adversary can derive about K. For this analysis, we allow adversary nodes to be active or passive.

3.2.4 Two New Schemes for MPKE Based on Reed-Solomon Codes

In this section, we propose two multi-path key establishment schemes, **Protocol 1** and **Protocol 2**. These protocols are obtained from a new PSMT protocol based on (n, k) RS codes. In **Protocol 1**, as in the HM scheme, A does not receive feedback from B. In **Protocol 2**, as in the JERT scheme, A receives feedback from B. For both schemes, we are interested in finding the optimal choice of (n, k) values such that B can recover a key with the desired entropy while A only transmits the minimum possible number of bits (i.e., the transmission overhead is optimized).

As mentioned above, our schemes are based on RS codes over finite field F_q. We assume that q is fixed and $q \geq n$ in the chosen (n, k) RS code.

Protocol 1

Here are the details of our first MPKE protocol, which is in fact a 1-round PSMT scheme if the parameters are chosen appropriately. We refer to this protocol as **Protocol 1**.

1. A chooses a random message $\mathbf{m} = (m_0, \ldots, m_{k-1}) \in F_q^k$ and encodes it into an RS codeword $\mathbf{c} = (c_0, \ldots, c_{n-1}) \in F_q^n$.

2. A sends the n codeword symbols over p pre-specified node-disjoint paths. Note that the number of symbols sent over any path is either $\lceil \frac{n}{p} \rceil$ or $\lfloor \frac{n}{p} \rfloor$.

3. B decodes the received symbols to a codeword $\mathbf{c}'$. Then a message $\mathbf{m}'$ is derived from $\mathbf{c}'$. Finally, a key K' is derived from $\mathbf{m}$ using a pre-specified key derivation function.

We discuss feasible and optimal choices of (n, k) values in Section 3.2.4.

Remark. When B receives a symbol c_i, B also needs to know its index i for decoding purposes (this applies to the HM scheme and JERT as well). This objective could be accomplished, for example, if A and B have some synchronization mechanism. In any event, we assume that B has some reliable means of knowing the index of any received symbol.

Analysis and Optimization Our goal is that the key K (derived from $\mathbf{m}$) has entropy ℓ symbols if $\mathbf{m}$ has entropy at least ℓ symbols. We will derive conditions to ensure that $\mathbf{m}$ has entropy at least ℓ symbols. Then the key derivation function is just a $(k, \ell, k-\ell, q)$-RF, obtained from Corollary 3.2.2. This would provide A and B with an unconditionally secure key in F_q^ℓ.

Suppose there are p node-disjoint paths from A to B and p_a of these paths are controlled by the adversary. Therefore the error rate is $e = p_a/p$. For simplicity, assume that n/p is an integer. Then in **Protocol 1**, $n/p \times p_a = ne$ symbols will be received by adversary nodes. These ne symbols may be altered or deleted.

First, we derive a condition to ensure *reliability* (i.e., so that B can correctly compute the key K). Since altering symbols makes it most difficult for B to recover $\mathbf{m}$, we assume that these ne symbols are all altered. For B to be able to correctly recover $\mathbf{m}$, the condition (3.8) becomes

$$ne \leq \frac{n-k}{2}. \tag{3.9}$$

Observe that (3.9) implies that $ne < n/2$, so $e < 1/2$.

Remark. If the scheme satisfies (3.9), then it is already a *prefectly reliable message transmission* scheme (for a definition, see [87]).

Now we consider *secrecy* of the transmitted message. In order for $\mathbf{m}$ to have entropy at least ℓ symbols, we require that

$$k - ne \geq \ell. \tag{3.10}$$

Using the fact that the desired entropy $\ell > 0$, it can be seen that (3.9) and (3.10) together imply that

$$\frac{k}{1-2e} \leq n < \frac{k}{e},$$

which yields $e < 1/3$.

So hereinafter we assume that $e < 1/3$. Under this assumption, (3.9) and (3.10) are equivalent to

$$\frac{k-\ell}{e} \geq n \geq \frac{k}{1-2e}. \tag{3.11}$$

The inequalities in (3.11) provide the conditions under which B can compute a key with entropy at least ℓ symbols. Note that (3.11) can equivalently be expressed as follows:

$$n(1-2e) \geq k \geq \ell + ne. \tag{3.12}$$

Suppose a value ℓ is fixed. Then we define an ordered pair (n, k) to be *e-feasible* if (3.11) (equivalently, (3.12)) is satisfied.

Given ℓ and e, the set of all e-feasible ordered pairs form a region, a typical example of which is indicated by the shadowed area in Figure 3.1. The optimal solution will be an ordered pair of integers (n, k) that is close to the ordered pair (n_{min}, k_{min}), which denotes the intersection of the two lines $k = n(1-2e)$and $k = \ell + ne$. It is easy to compute

$$n_{min} = \frac{\ell}{1-3e} \qquad \text{and} \qquad k_{min} = \frac{\ell(1-2e)}{1-3e}. \tag{3.13}$$

Clearly $n_{min} > 0$ and $k_{min} > 0$ because $e < 1/3$. n_{min} represents the optimal transmission size in the protocol.

Theorem 3.2.3. *Suppose ℓ is a positive integer and $0 \leq e < 1/3$. Suppose (n, k) is e-feasible. Finally, suppose that there are p disjoint paths from A to B, where $p_a = pe$ of these paths are controlled by the adversary. Suppose that n/p is an integer. Then* **Protocol 1** *yields an ℓ-secure secret key. The total transmission of* **Protocol 1** *consists of n symbols.*

If we apply **Protocol 1** with $(n, k) = (n_{min}, k_{min})$, then the *transmission overhead* is

$$\frac{n}{\ell} = \frac{1}{1-3e} = \frac{p}{p-3p_a},$$

which is optimal, by (3.6).

Protocol 1 is analyzed in terms of the error rate e. In general, the error rate will not be known. In practice, Alice and Bob would choose a value $e^* < 1/3$ which they hope is an upper bound on e. They would then execute **Protocol 1** with an e^*-feasible ordered pair (n, k). It is easy to see that an e^*-feasible ordered pair is also e-feasible provided that $0 \leq e \leq e^*$, so **Protocol 1** will still work correctly in these circumstances.

Example. Suppose that $e = 1/5$, $p = 5$ and $\ell = 40$. Then we can take $n = 100$ and $k = 40$ in Theorem 3.2.3. That is, we obtain a 40-secure key using a $(100, 40)$ RS code under the assumption that at most one of five node-disjoint paths joining A and B is controlled by the adversary.

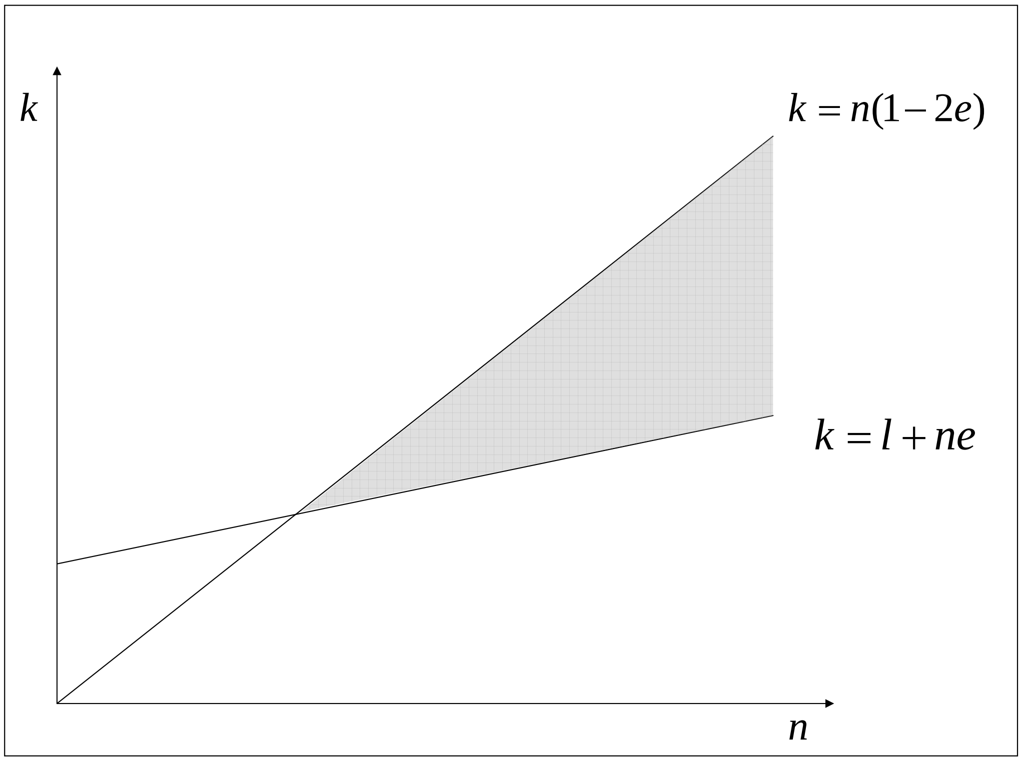


Figure 3.1: e-feasible ordered pairs (n, k) for given error rate e and desired key entropy ℓ.

Protocol 2

In **Protocol 1**, if the actual error rate $e < e^*$, then the protocol might transmit more information than is actually necessary; i.e., the efficiency might not be optimal. To reduce the number of transmitted symbols, A can use feedback from B. This idea was first proposed in JERT [32]. JERT is designed for two neighbouring nodes that can communicate directly. It is assumed that the channel connecting A and B is a broadcast channel. Therefore, it provides data integrity, but no confidentiality or data origin authentication. The lack of confidentiality or authentication is not a problem, as this channel is used only for message authentication. We assume a similar channel in our protocol.

Next we define **Protocol 2**, where B will send feedback to A using the broadcast channel. Let e^* be the maximum error rate that the protocol is designed for (i.e., an e^*-feasible ordered pair (n, k) is chosen for use in the protocol). As before, assume that there are p node-disjoint paths from A to B and assume for convenience that $p \mid n$. In **Protocol 2**, MAC denotes a secure message authentication protocol.

1. A chooses a random message $\mathbf{m} = (m_0, \ldots, m_{k-1}) \in F_q^k$ and encodes it into an RS codeword $\mathbf{c} = (c_0, \ldots, c_{n-1}) \in F_q^n$.

2. In each of n/p *rounds*, A sends one codeword symbol over each of the p pre-specified node-disjoint paths.

3. After each round, B attempts to decode the symbols he has received in the current and all previous rounds to a codeword $\mathbf{c}'$. If he is successful, then a message $\mathbf{m}'$ is derived from $\mathbf{c}'$ and a key K' is derived from $\mathbf{m}'$ using the key derivation function.

4. If B is able to compute a (possible) key K', then B start a conventional message authentication code (MAC) based mutual authentication protocol with A over the broadcast channel. In the protocol, A and B verify if the peer holds the same key. If the authentication succeeds, then both A and B stop.

   Since we assumed that the broadcast channel provides integrity, the adversary is not allowed to change the messages between A and B in the authentication protocol. If A and B have the same key, then the adversary is not able to prevent the authentication from succeeding. If the adversary does not know the key held by A, it cannot stop A from sending. If the adversary does not know the key held by B, it cannot stop B from receiving, either.

Remark. This protocol is also secure against mobile adversaries. (A *mobile adversary* is allowed to compromise different nodes in different rounds, subject to the constraint that the error rate is at most e^* in any given round.)

Protocol 2 is computationally secure whenever MAC is a computationally secure message authentication code. It would be possible to analyze **Protocol 2** in the setting of unconditional security, by employing an unconditionally secure message authentication code. This analysis would be required to take into account the fact that usage of a key in an unconditionally secure MAC "leaks" information about the key. However, as noted in Section 3.2.1, a "practical" MPKE scheme will not be unconditionally secure, so we do not pursue this theme further here.

Analysis First, let us consider the properties of security and reliability. It is certainly possible that B computes an incorrect key, but he will not accept a wrong key (except with very small probability) due to the mutual authentication protocol. Eventually, after some number of rounds, B will be able to compute the correct key provided that $e \leq e^*$ (for details, see below). Therefore **Protocol 2** achieves reliability. Secrecy follows from the same analysis as for **Protocol 1**.

Next, we analyze the efficiency of **Protocol 2** by determining the number of rounds required for B to be able to compute the correct key K. After r rounds, B has received rp symbols, at most rpe of which have been altered. The number of symbols which have not yet been transmitted to B is $n - pr$. Thus B has a received vector in which $\epsilon \leq per$ and $\delta = n - pr$. Referring to (3.8), B can correctly decode this received vector if

$$2rpe + n - pr \leq n - k,$$

which is equivalent to

$$r \geq \frac{k}{p(1-2e)}. \tag{3.14}$$

Therefore the correct key is computed by B after at most $\lceil k/(p(1-2e)) \rceil$ rounds. It follows that the *speedup factor* of **Protocol 2** as compared to **Protocol 1** is

$$\frac{1-2e^*}{1-2e}.$$

If $e = e^*$, then the number of rounds required is $\lceil k/(p(1-2e^*)) \rceil$. If $e = 0$, then the number of rounds required is $\lceil k/p \rceil$.

Summarizing the above discussion, we have the following theorem.

Theorem 3.2.4. *Suppose ℓ is a positive integer and $0 < e \leq e^* < 1/3$ and suppose (n, k) is e^*-feasible. Suppose that there are p disjoint paths from A to B, where $p_a = pe$ of these paths are controlled by the adversary. Then* **Protocol 2** *yields an ℓ-secure secret key. The total transmission of* **Protocol 2** *consists of (roughly) $n(1-2e^*)/(1-2e)$ symbols.*

Remark. In practice, B would not attempt to decode the received vector after every round. The exact error rate $e = p_a/p$, where $p_a \leq pe^*$ is an integer. Using (3.14), we see that it is sufficient for B to decode a received vector only when a round r has the form

$$r = \left\lceil \frac{k}{p\left(1 - \frac{2i}{p}\right)} \right\rceil = \left\lceil \frac{k}{p - 2i} \right\rceil$$

for some integer $i \geq 0$.

3.2.5 A Scheme Tolerating Error Rate $< 1/2$

Both of our protocols described in Section 3.2.4 assume that the number of paths controlled by the adversary is less than a 1/3 fraction of the number of paths connecting A and B. A higher fraction (less than 1/2) of paths controlled by the adversary could be tolerated by using appropriate message transmission schemes mentioned in Section 3.2.1. These schemes either require additional rounds of communication or they are not perfectly reliable. They are also somewhat complicated and/or inefficient. In this section, we present a new 2-round protocol for a weakened version of message transmission that is very simple and efficient, and well-suited for application as a MPKE scheme. Our protocol will be computationally secure provided that certain specified ingredients exist.

Our scheme has the following properties:

- We assume that A and B are joined by p node-disjoint paths, at most p_a of which are controlled the adversary, where $p \geq 2p_a + 1$.
- We require a mapping $h : \mathcal{K} \to \mathcal{T}$, where $\mathcal{K}$ is the *key component space* and $\mathcal{T}$ is the *tag space*. We will take $\mathcal{K} = F_q$ for some prime power $q \geq p$.
- The scheme is perfectly reliable if h is injective (in this case, the scheme enables A and B to establish a shared key $K \in \mathcal{K}$ with probability equal to 1 independent of any computationl assumptions). The scheme is (computationally) reliable if h is second-preimage resistant.
- Under the assumption that h is a random function, the scheme provides secrecy of the established key. (For a random function, a computationally-bounded adversary is unable to compute any non-negligible information about a secret value L, when the adversary is given only the value $h(L)$.)
- The scheme is a two-round scheme.

Here is the protocol, which we term **Protocol 3**.

1. For $1 \leq i \leq p$, A chooses a *key component* $L_i \in \mathcal{K}$ independently and uniformly at random. Then A computes $h_i = h(L_i)$, $i = 1, \ldots, p$.
2. For $1 \leq i \leq p$, A sends L_i over the ith path. Also, for $1 \leq i, j \leq p$, $i \neq j$, A defines $h_{i,j} = h_i$ and sends $h_{i,j}$ over the jth path.

3. (a) For $1 \leq i \leq p$, B computes

$$\mathbf{check}(i) = \{j : h_{i,j} = h(L_i)\}.$$

   (b) B *accepts* L_i if and only if $|\mathbf{check}(i)| \geq p - p_a - 1$.

   (c) B defines

$$\mathbf{accept} = \{i : B \text{ accepts } L_i\}$$

   and $n = |\mathbf{accept}|$.

   (d) B defines $\mathbf{m} = (L_i : i \in \mathbf{accept})$.

   (e) B computes $K = f(\mathbf{m})$, where f is an $(n, p - p_a, n - (p - p_a), q)$-resilient function.

4. B transmits **accept** to A over every one of the p paths.

5. (a) A determines **accept**, as it will be correctly received over at least $p - p_a$ paths (i.e., a majority of the paths).

   (b) A computes $\mathbf{m}$ and K exactly as B did.

Remark. As described above, **Protocol 3** is not a message transmission scheme due to the fact that the value of the derived key, K, is not specified *a priori*; its value depends on possible actions of the adversary. This is sufficient for the goals of an MPKE scheme. However, if desired, it is easy to use a standard trick to make a minor alteration to our scheme in order to transmit a predetermined key K^* from A to B. Namely, the protocol would be initiated by B (instead of A), and in the second round, A would send $K^* + K$ to B along with **accept**.

Remark. In practice, we could take h to be a second preimage-resistant and one-way hash function. Another alternative is to let h be a semantically secure public-key cryptosystem with randomly chosen public key, in which case h would be injective.

Analysis

First, we show that if B accepts a key component L_i, then it was not altered by the adversary. Suppose that the adversary replaces L_i by a different value L_i'. Assuming that h is injective, we have that $h(L_i') \neq h(L_i)$. In order for the adversary to make B accept L_i' (in step 3(b)), he would have to change at least $p - p_a - 1$ of the $p - 1$ values $h_{i,j}$ $(j \neq i)$. But if the adversary controls p_i. then the adversary controls at most $p_a - 1$ of the other $p - 1$ paths. We have $p_a - 1 < p - p_a - 1$ because $p \geq 2p_a + 1$. Therefore, in this situation, the scheme is perfectly reliable.

If h is not injective but it is second-preimage resistant, then a computationally-bounded adversary is unable to find L_i' such that $h(L_i') = h(L_i)$, even if such L_i' exist. The scheme is (computationally) reliable in this case.

It remains to evaluate the secrecy of the derived key K. First, we observe that if the adversary does not control the ith path, then he cannot determine any information about the key component L_i. This is because we are assuming that h is a random mapping. Now, let r denote the number of rejected key components; $r = p - a$. Any rejected key component lies on a path controlled by the adversary. Therefore the number of accepted key components that lie on paths controlled by the adversary is at most $p_a - r = n - (p - p_a)$. Now, the n-tuple $\mathbf{m}$ contains at most $n-(p-p_a)$ components that are known to the adversary. Hence, application of a $(n, p-p_a, n-(p-p_a), q)$-resilient function will yield a key whose entropy is $p - p_a$ symbols. This resilient function exists by Corollary 3.2.2.

Finally, A is also able to compute K because A is able to correctly determine the set **accept** after receiving p copies of it from B (at most p_a of these copies are altered by the adversary, so the correct **accept** can be determined by majority rule).

Let's next analyze the transmission overhead of the scheme. For the purpose of this analysis, assume that $|\mathcal{K}|$ is $\Theta(|\mathcal{T}|)$. Then the derived key has entropy $p - p_a$ symbols and the total transmission from A to B is $\Theta(p^2)$ symbols. The total transmission from B to A is p^2 bits (the set **accept** can be represented as a bitstring of length p). Since $q \geq p$, this is at most p symbols. So the total transmission is $\Theta(p^2)$ symbols, and the transmission overhead is at most

$$\Theta\left(\frac{p^2}{p-p_a}\right).$$

Since $p - p_a > p/2$, the transmission overhead is $\Theta(p)$.

Theorem 3.2.5. *Suppose ℓ is a positive integer and $0 \leq e < 1/2$. Suppose that there are p disjoint paths from A to B, where $p_a = pe$ of these paths are controlled by the adversary. Then* **Protocol 3** *yields a $(p-p_a)$-secure secret key. The total transmission of* **Protocol 3** *consists of $\Theta(p^2)$ symbols, and the transmission overhead is $\Theta(p)$.*

Chapter 4

RFID Authentication Protocols

In this chapter, we study RFID authentication protocols. In Section 4.1, we present a new polynomial based RFID authentication protocol in Section 4.1.1, and give its security analysis in 4.1.2 and performance analysis in 4.1.3.

In Section 4.2, we review short padding attacks and secure paddings in Section 4.2.1, then we analyze WIPR in Section 4.2.2. In Section 4.2.3, we present WIPR-SAEP, a WIPR variant secured using SAEP padding, and WIPR-RNS, a WIPR variant based on residue number systems which uses fewer hardware gates.

In Section 4.3, we analyze an RFID identification scheme, and show how to ensure forward and backward untraceability of RFID authentication protocols by using a robust PRBG.

4.1 Polynomial Based Protocol

4.1.1 Scheme Description

In this section, we present our RFID protocol. We assume that there are N tags and one reader (here the reader refers to an entity consisting of the actual reader, the database that stores the tag information, and perhaps an application server which processes the data). We also assume that a tag is capable of generating random bits, computing cryptographic hash functions, and modular multiplication over a field F_{2^l}. Such tags would fall into the category of symmetric-key tags in [61].

Initial Setup

The initial setup configures the reader and the tags as follows.

- Choose an l-bit block cipher and a secret key. Let E be the encryption function and D be the decryption function.
- Generate m random bivariate polynomials $f_1(x, y), \ldots, f_m(x, y)$ over a finite field F_{2^l}. The degree of x and y in each f_i is at most k.
- Assign the m polynomials to the reader.
- For tag $i, 1 \leq i \leq N$, compute $x_i = E(i)$ as its meta ID, then
  - store m univariate polynomials $f_{1,i}(y) = f_1(x_i, y), \ldots, f_{m,i}(y) = f_m(x_i, y)$ in the tag in a randomly permuted order.
  - Set a counter c to 0.
  - Set an interrogation threshold Q_{max}, which is the maximum number of queries that the tag will answer correctly.
  - Set a number b, which will be used in the protocol.
- Choose a secure hash function $h : \{0, 1\}^* \mapsto \{0, 1\}^l$ which will be used by readers and tags.

Note that the meta ID x_i is not stored in the tag or the reader. We will discuss the parameter selection for l, m, b, k, and Q_{max} in the analysis and performance sections.

Authentication

The authentication process between a reader and a tag i is as follows.

1. The reader generates $r \in F_{2^l}$ uniformly at random, then sends r to the tag.
2. If $c > Q_{max}$, then the tag responds with b random points in $F_{2^l} \times F_{2^l}$. Otherwise, the tag does the following:
   (a) Generate $r' \in F_{2^l}$ uniformly at random, and compute $y' = h(r||r')$.
   (b) Choose g from $\{f_{1,i}, \ldots, f_{m,i}\}$ randomly, and compute $z' = g(y')$.
   (c) Generate $b-1$ pairs of points $(r_1, z_1) \ldots, (r_{b-1}, z_{b-1}) \in F_{2^l} \times F_{2^l}$ uniformly at random.
   (d) Send the b points $(r', z'), (r_1, z_1), \ldots, (r_{b-1}, z_{b-1})$ to the reader in a random order.
   (e) Set $c = c + 1$.
3. After receiving the b points, for each point (r', z') and each polynomial $f \in \{f_1, \ldots, f_m\}$, the reader solves the equation $z' = f(x, h(r||r'))$. The reader needs to solve mb such equations, and each equation generates up to k solutions. If any solution x is a valid meta ID; i.e., $1 \leq D(x) \leq N$, then the reader identifies the tag ID as $D(x)$.

Remark. In Step 2 (b), to pick a polynomial, the tag can use two different approaches. One is *random choice* and the other is *random permutation*. In random choice, each time, the tag chooses one polynomial from the m polynomials uniformly at random. In random permutation, the choice of g in m consecutive authentication sessions is a random permutation of the m polynomials assigned to the tag. In the following parts, we assume the random permutation approach.

Correctness

If the reader receives b points from a valid tag with ID i, then one of the points (r', z') will satisfy $z' = f(E(i), h(r||r'))$ for a polynomial $f \in \{f_1, \ldots, f_m\}$. At the reader side, $E(i)$ will be the solution of x to the equation $z' = f_j(x, h(r||r'))$. So the valid tag will be correctly identified.

The server will also get up to $mbk - 1$ other solutions in one authentication session. The distribution of the meta IDs can be considered to be uniformly at random in F_{2^l}, so the probability that one or more of these solutions happen to be valid meta IDs is estimated to be

$$1 - \left(1 - \frac{N}{2^l}\right)^{mbk-1}. \tag{4.1}$$

4.1.2 Security Analysis

In this section, we analyze the security and privacy properties of our protocol. Recall that in the attack model, the adversary can eavesdrop and query any tag, compromise some tags, reset the tags, and change the messages between a tag and a reader. The goal of the adversary is to impersonate or trace the uncompromised tags.

Query and Recovery

First, we consider the query-and-recovery attack. In this attack, the adversary repeatedly queries a tag, collects the responses, and then tries to recover the polynomials assigned to the tag. This is an intermediate step toward impersonation and tracing.

Our security analysis for the query-and-recovery attack is based on the hardness of the *noisy polynomial interpolation problem*, which is related to the well-known *polynomial reconstruction problem*. Next we present existing results about these problems in the literature, followed by our analysis.

Preliminary First we review the polynomial reconstruction (PR) problem [63] and the noisy polynomial interpolation (NPI) problem [81].

Definition 4.1.1. *(Polynomial reconstruction)*

Input: Integers k and t, and T points $\{(x_i, y_i) : i \in [1, T]\}$ where $x_i, y_i \in F_{2^l}$.

Output: All univariate polynomials P of degree at most k such that $P(x_i) = y_i$ for at least t values $i \in [1, T]$.

The fastest algorithm known to solve this problem is the Guruswami-Sudan algorithm [52]. It solves the problem when $t > \sqrt{Tk}$ in time polynomial in T. When $t \leq \sqrt{Tk}$, the current state of knowledge suggests that the problem may be difficult even in the light of recent extensions of list decoding or average case decoding for related families of codes [63].

A variant of the PR problem, denoted as the noisy polynomial interpolation (NPI) problem, is as follows:

Definition 4.1.2. *(Noisy Polynomial Interpolation)*

Input: S sets of points generated as follows: Pick a random polynomial P over F_{2^l} of degree at most k. Generate $S \geq k+1$ sets, each containing B points. The x coordinate of each point is randomly chosen from F_{2^l} subject to the condition that all x values are distinct and different from 0. In each set there is exactly one point (x, y) which satisfies $y = P(x)$. For the other $B-1$ points, the y coordinate is chosen randomly.

Output: the polynomial P.

The NPI problem is presented in [81]. A previous version of this problem was presented in [80] in which the points in the same set all have the same x coordinate. In [15], Bleichenbacher and Nguyen show that having the same x coordinate allows meet-in-the-middle attacks and lattice attacks. However, it is unknown how to employ these attacks against NPI. It appears that, although there is more information given in NPI than in the PR problem, NPI may be as hard as the PR problem.

Parameter Selection for NPI In [81], Noar and Pinkas also proposed a cryptographic assumption based on the NPI problem. They assumed that, if S, B and k are polynomial in the security parameter and $S < \sqrt{SBk}$, then the problem is hard. Here we further analyze the security level under concrete parameter settings.

First we review the analysis of parameter choices for the PR problem in [62]. In [62], the best approach to solve the PR problem is assumed to be one of the two choices: (1) choose $k+1$ points to recover a polynomial and then test all $\binom{T}{k+1}$ polynomials, or (2) delete d points, where $t > \sqrt{(T-d)k}$, and use the GS algorithm on the remaining $T-d$ points. We call the first approach *exhaustive search*; its idea is clear. The idea of the second approach is to delete d points. If all the d points are not on the polynomial P, then it happens that $t > \sqrt{(T-d)k}$ so that GS will output the proper P. We call this approach *exhaustive deletion.*

We note that the idea of exhaustive search and exhaustive deletion can be generalized to selecting n points where $k+1 \leq n \leq T$. We use this generalized idea for the NPI problem and consider the following new probabilistic algorithm A as shown in Algorithm 7 to solve NPI.

```
choose $n$ sets of points
choose $\beta$ points from each set
use the $n\beta$ points as input to the GS algorithm, and output all polynomials of degree
at most $k$ that fit at least $k+1$ points.
```

Algorithm 7: A

Note that A may output more than one polynomial. If P is among them, we consider A successful in solving the problem.

Let $P_{NPI}(S, B, k, n, \beta)$ be the probability that P will be outputted. Let t be the number of points in the set of $n\beta$ points that satisfy $y = P(x)$. It is clear that $0 \leq t \leq n$ and t follows a binomial distribution with parameters $(\frac{\beta}{B}, n)$. If $t > \sqrt{n\beta k}$, then P will be outputted. It holds that

$$\begin{aligned} P_{NPI}(S, B, k, n, \beta) &= \Pr[t > \sqrt{n\beta k}] \\ &= \sum_{i=\lfloor\sqrt{n\beta k}\rfloor+1}^{n} \Pr[t = i] \\ &= \sum_{i=\lfloor\sqrt{n\beta k}\rfloor+1}^{n} \binom{n}{i} \left(\frac{\beta}{B}\right)^{i} \left(1 - \frac{\beta}{B}\right)^{n-i}. \end{aligned} \tag{4.2}$$

Remark. Note that, when $\beta = 1$ and $n = k + 1$, the algorithm becomes an exhaustive search and

$$P_{NPI}(S, B, k, k + 1, 1) = \left(\frac{1}{B}\right)^{k+1}.$$

When $\beta = B$, the algorithm is deterministic for given n. In this situation, if $n \leq Bk$, then $\lfloor\sqrt{nBk}\rfloor + 1 > n$ so that $P_{NPI}(S, B, k, n, B) = 0$. If $n > Bk$, then

$$P_{NPI}(S, B, k, n, B) = \sum_{i=\lfloor\sqrt{nBk}\rfloor+1}^{n} \binom{n}{i} 0^{n-i} = 1.$$

Regarding the hardness of NPI, we make the following assumption to estimate the concrete security level.

Assumption 4.1.3. *Let*

$$Adv_{S,B,k}^{NPI} = \max\{P_{NPI}(S, B, k, n, \beta) : 1 \leq \beta \leq B, k + 1 \leq n \leq S\}. \tag{4.3}$$

Let n_0, β_0 be the optimal choices of n, β for A to achieve $Adv_{S,B,k}^{NPI}$. Let τ be the running time of A using n_0, β_0. We assume that no algorithm can solve NPI with probability greater than $Adv_{S,B,k}^{NPI}$ using time at most τ.

$Adv_{S,B,k}^{NPI}$ can be estimated as

$$Adv_{S,B,k}^{NPI} = \max\left\{\left(\frac{1}{B}\right)^{k+1}, P_{NPI}\left(S, B, k, S, \left\lceil\frac{S}{k}\right\rceil - 1\right)\right\} \tag{4.4}$$

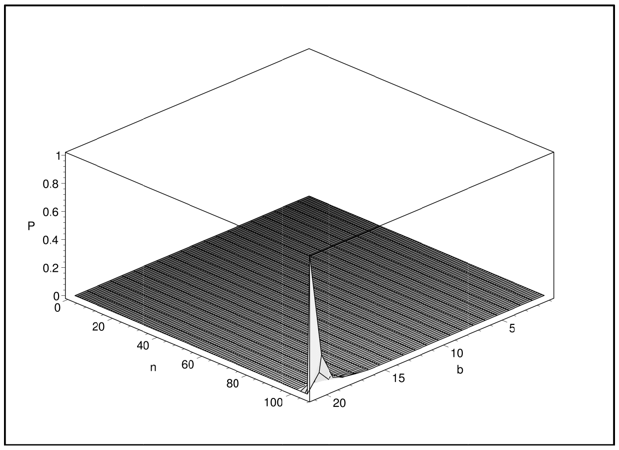


Figure 4.1: P_{NPI} when $S \geq Bk$

.

Discussion on NPI and Exhaustive Search and Deletion. We discuss the optimal choices of n and β for A to achieve $Adv_{S,B,k}^{NPI}$. It is difficult to give an analytic result, so here we discuss this question based on some simulations.

Figure 4.1 shows $P_{NPI}(S, B, k, n, \beta)$ as a function of n and β for $1 \leq n \leq Bk, 1 \leq \beta \leq B$, $B = 21$ and $k = 5$. We see that when n approaches Bk and β approaches B, P_{NPI} increases sharply to 1. That indicates that if there are enough sets of points ($s \geq Bk$), then the adversary can easily recover the polynomial.

Figure 4.2 shows $P_{NPI}(S, B, k, n, \beta)$ as a function of n and β for $1 \leq n \leq S, 1 \leq \beta \leq B$, $B = 5$, $k = 5$ and $S = 83 \leq Bk$. There are two points that have peak values among their vicinities in the graph. One is at $n = k + 2$ and $\beta = 1$, the other is at $n = 83 = S$ and $\beta = 13$.

We note that, if we use the exhaustive search approach, we will choose $n = k + 1$ and $\beta = 1$, and then we have $P_{NPI}(S, B, k, k + 1, 1) = 0.1 \times 10^{-7}$ which is lower than the peak value where $P_{NPI}(S, B, k, k + 2, 1) = 0.8 \times 10^{-7}$, but the difference is small in view of their magnitudes. Also, the point $(n = k + 1, \beta = 1)$ for exhaustive search is close to that for the peak value at $(n = k + 2, b = 1)$.

If we take the exhaustive deletion approach, then we will choose $n = S = 83, \beta = \lceil \frac{S}{k} \rceil - 1 = 16$, and have $P_{NPI}(S, B, k, 83, 16) = 0.4 \times 10^{-8}$ which is lower than the other peak value $P_{NPI}(S, B, k, 83, 15) = 0.4 \times 10^{-7}$, but difference is small in view of their magnitudes. Also the point for exhaustive search $(n = 83, \beta = 16)$ is close to that for the peak value at

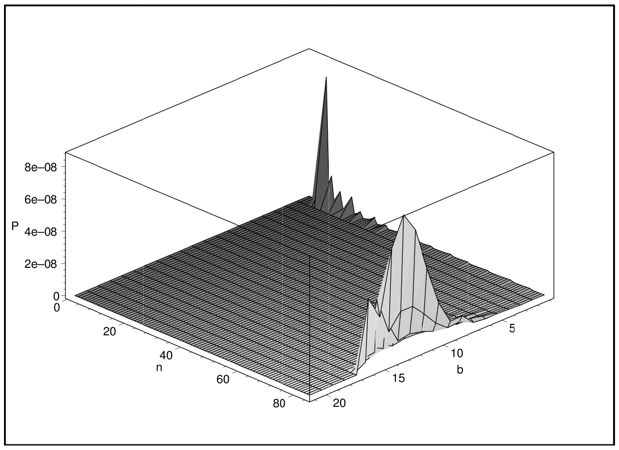


Figure 4.2: P_{NPI} when $S < Bk$

.

$(n = 83, \beta = 15)$.

The above observation leads to the conjecture that, for $S < Bk$, the results of exhaustive search and exhaustive deletion are close to the best strategy for the adversary. Then the advantage of the adversary may be estimated as

$$Adv^{NPI}_{S,B,k} = \max\left\{\left(\frac{1}{B}\right)^{k+1}, P_{NPI}\left(S, B, k, S, \left\lceil\frac{S}{k}\right\rceil - 1\right)\right\}. \tag{4.5}$$

Security Under Query-and-Recovery Now we relate the difficulty of the query-and-recovery attack to the difficulty of solving the NPI problem.

Recall that in the protocol, to answer a challenge r, the tag computes $z' = g(h(r||r'))$ where r' is a random number generated by the tag. $h(r||r')$ can be considered as random to the adversary. This point (z', r') is sent along with $b - 1$ other random points. In a query-and-recovery attack, the adversary queries a tag Q times. In every consecutive m queries, the tag uses each of its m polynomials once and in a random order. The problem for the adversary is to recover these polynomials after Q queries. We have the following result.

Theorem 4.1.4. *Suppose Assumption 4.1.3 holds. If the adversary queries a tag Q times (where we assume that $m|Q$), then the probability that it can recover any polynomial of the tag within time τ is at most*

$$\frac{1}{m} Adv^{NPI}_{Q/m, mb-m+1, k}. \tag{4.6}$$

Proof. We reduce an NPI problem with parameters ($S = Q/m, B = mb - m + 1$) to a query-and-recovery problem. We choose $m - 1$ random polynomials. For each polynomial, we generate Q/m points from Q/m random x values, and put the Q/m points in the Q/m sets. This becomes a query-and-recovery problem of Q queries, m polynomials of degree k, and b points in each answer, and with probability $1/m$, the polynomial recovered by the adversary is the answer to the NPI problem. $\square$

Given ϵ, m, b, and k, there is a Q_{max} such that for $Q \leq Q_{max}$, it holds that

$$Adv^{NPI}_{Q/m,mb-m+1,k} \leq \epsilon.$$

Therefore, Q_{max} is the maximum number of queries allowed for one tag. Q_{max} can be estimated as

$$Q_{max} \approx ((b-1)m^2 + m)k \tag{4.7}$$

Equation (4.7) indicates that Q_{max} increases linearly with m^2 and k. mk indicates the memory overhead to store the m polynomials of degree k in a tag. Q_{max} also increases linearly with b, which indicates the communication overhead and the number of random points the tag needs to generate.

Compromise and Recovery

In this attack, the adversary compromises some tags, obtains the polynomials assigned to these tags, and tries to recover the bivariate polynomials used in the server side. This is an intermediate step toward impersonation and tracing.

First we express the polynomial assignment in the form of a matrix computation. Let

$$X = \begin{pmatrix} 1 & x_1 & \dots & {x_1}^k \\ & \ddots & & \\ & & \ddots & \\ 1 & x_n & \dots & {x_n}^k \end{pmatrix},$$

where $x_1, \dots, x_n$ are the meta IDs assigned to the n tags. Let $M_1, \dots, M_m$ be $(k+1) \times (k+1)$ matrices, which are matrix representations of the bivariate polynomials $f_1, \dots, f_m$. Let $Y_i = XM_i$. Then Y_i is an $n \times (k+1)$ matrix, and row j of M_i corresponds to the univariate polynomial generated using f_i and assigned to tag j.

We use $M[r]$ to denote the r^{th} row of a matrix M, and $M[r][c]$ to denote the entry at the r^{th} row and c^{th} column of M. It is clear that $Y_j[i] = X[i]M_j$, and the univariate polynomials assigned to tag i are $Y_1[i], \dots, Y_m[i]$.

For the adversary to recover an M_i, it is necessary to know Y_i. Suppose the adversary obtains Y_i. He then needs to solve the following system of $n(k+1)$ equations with $n+(k+1)^2$ unknown variables:

$$Y_i[r][c] = \sum_{j=0}^{k} {x_r}^j M_i[j][c], \tag{4.8}$$

where $0 \leq r \leq n-1, 0 \leq c \leq k$.

A necessary (but maybe not sufficient) condition to solve (4.8) is $n \geq \frac{(k+1)^2}{k}$. We assume that when $n \geq \frac{(k+1)^2}{k}$, the adversary can solve (4.8) in a practical time period τ.[1]

However, when $Y_1[i], \ldots, Y_m[i]$ are assigned to the tag x_i, their order is randomly permuted. So by compromising n tags, the adversary does not know which n polynomials in the n tags are generated from the same M. We assume that the best he can do is to draw one polynomial from each tag as a row of Y_i, and solve $x_1, \ldots, x_n$ and M_i in (4.8). With probability $1/m^{n-1}$, the polynomials in Y_i are generated from the same bivariate polynomial f. Therefore, we estimate that the probability that the adversary can compute one M_i in time τ is

$$\begin{aligned} Adv_{CR} &= \frac{1}{m^{n-1}} \\ &\leq \frac{1}{m^{\left\lceil \frac{(k+1)^2}{k} \right\rceil - 1}} \\ &= m^{-k-2}. \end{aligned} \tag{4.9}$$

Remark. After the adversary has compromised n tags, if he also knows the meta IDs $x_1, \ldots, x_n$ of the tags, then he can use the GS algorithm to recover $M_1, \ldots, M_m$. However, when $x_1, \ldots, x_n$ are unknown, the GS algorithm is no longer applicable, and we assume that there is no efficient algorithm to solve the problem. It is worthwhile to further investigate the validity of this assumption.

Man-in-the-middle Attacks

In this attack, the adversary can modify the messages between the tag and the reader. We consider the following specific attack. The adversary modifies one of the b points returned

[1](4.8) is a system of nonlinear polynomial equations. For general systems of nonlinear polynomial equations where the number of unknown variables u equals the number of equations v, there is no efficient algorithm. If the system is overdefined; i.e., $v > u$, then the linearization technique may sometimes be used to solve the problem efficiently [29]. But for (4.8), the linearization technique does not work. On the other hand, we cannot rule out the possibility that the special form of (4.8) may make some efficient algorithms possible. Here we assume that (4.8) can be solved efficiently. This assumption may underestimate the security of the protocol.

by the tag. If the reader rejects the tag, then the adversary knows that the modified point is on one of the tag's polynomials. After repeating the attack $mb(k+1)$ times, the adversary gathers $(k+1)$ of points (on average) for each of the polynomials of a tag. However, the adversary does not know which $k+1$ points are for the same polynomial. If the adversary randomly chooses $k+1$ points, then the probability that they are on the same polynomial is

$$\frac{m}{\binom{m(k+1)}{k+1}}. \tag{4.10}$$

If the adversary tries to use the GS algorithm to solve the problem, then it needs to repeat the attack q times where

$$\frac{q}{bm} > \sqrt{\frac{qk}{b}}.$$

It requires that

$$q > m^2 bk. \tag{4.11}$$

Reset Attacks

In this attack, the adversary can reset a tag's internal state. This attack may be effective if a tag uses a pseudorandom number generator, and the pseudorandom number generator repeats its output when its internal state is reset. When the tag uses a true random number generator, or a pseudorandom number generator with certain properties, then the reset attack does not work (see Section 4.3 for examples). In this protocol, we assume that proper pseudorandom number generator is used and the protocol is secure under reset attacks.

Impersonation

In the impersonation attack, the adversary tries to impersonate a uncompromised tag to a reader. In the RFID literature, this is often named counterfeiting or cloning. Since cloning implies replicating a tag, we think that impersonation may describe the goal of the adversary better. For example, the adversary may combine the information gathered from a tag to generate new messages to impersonate a tag, or use the information gather from one tag to impersonate another tag. In both cases, the adversary tries to cheat the reader in a way other than by replicating. After receiving a challenge r, to make the reader accept it as a valid tag, the adversary A needs to generate a point (r', z') such that $z' = g(h(r||r'))$ where g is a polynomial assigned to the tag. As we have analyzed, the adversary cannot learn the polynomials assigned to a tag by querying a tag or compromising other tags. Since A does not know g, he cannot compute z' by evaluating $g()$. Then A may choose a z' previously generated by the tag. A can do this in a probabilistic way: after observing a query and response from the tag, A randomly picks one point from the response. With probability $1/b$, A gets r_0, r'_0, and z'_0 where $z'_0 = f(h(r_0||r'_0))$. Then A needs to find r' such that $h(r||r') = h(r_0||r'_0)$. When r is a random challenge, and the hash function $h()$ is modelled as a random oracle, then the probability that A chooses r such that $h(r||r') = h(r_0||r'_0)$ is

$$\begin{aligned} & \Pr[h(r \parallel r') = h(r_0 \parallel r'_0)] \\ = \ & \Pr[h(r \parallel r') = h(r_0 \parallel r'_0) \wedge r = r_0] + \Pr[h(r \parallel r') = h(r_0 \parallel r'_0) \wedge r \neq r_0] \\ \leq \ & \Pr[r = r_0] + \Pr[h(r \parallel r') = h(r_0 \parallel r'_0)|r \neq r_0] \\ = \ & \frac{1}{2^l} + \frac{1}{2^l} \\ = \ & \frac{1}{2^{l-1}}. \end{aligned} \tag{4.12}$$

A may just send random points and hope that at least one of them satisfies $z' = f(x', h(r||r'))$ for a polynomial $f \in \{f_1, \ldots, f_m\}$ and a valid meta ID x'. The probability that this happens is similar to (4.1).

Tracing

Now we consider the tracing problem. Suppose that the adversary observes or participates in two authentication sessions as a reader. The problem for the adversary is to tell if the two sessions involve the same tag.

The output of a tag is b points. $b - 1$ of them are random points, and only one point $(r', g(h(r||r')))$ contains the information related to the tag. As we have analyzed, the adversary cannot learn the polynomials of an uncompromised tag. Without knowing g, only when $y_1 = h(r||r'_1) = y_2 = h(r||r'_2)$, can the adversary tell that two points $(y_1, g(y_1))$ and $(y_2, g(y_2))$ are generated by the same polynomial. When r'_1 and r'_2 are generated by the tag at random, and the hash function $h()$ is modelled as a random oracle, the probability that the adversary can trace a tag by its response is

$$\begin{aligned} \Pr[h(r||r'_1) = h(r||r'_2)] \ & \leq \ \Pr[r'_1 = r'_2] + \Pr[h(r||r'_1) = h(r||r'_2)|r'_1 \neq r'_2] \\ & = \ \frac{1}{2^l} + \frac{1}{2^l} \\ & = \ \frac{1}{2^{l-1}}. \end{aligned} \tag{4.13}$$

Note that if a tag has been queried more than Q_{max} times, and the adversary can get more information than the tag's response (i.e., if an authentic reader accepts the tag), then the adversary may trace the tag.

4.1.3 Performance

We discuss the performance of the protocol under a concrete parameter setting. We set $l = 64, m = 16, k = 8, b = 8$, and $Q_{max} = 13700$.

Security and Privacy

The security and privacy provided by the protocol is as follows.

- The adversary queries a tag and tries to recover the secret polynomials held by the tag. In each trial, the probability that the adversary can succeed is at most 2^{-56} (by (4.6)).
- The adversary compromises several tags and tries to determine the secret bivariate polynomials held by the server. In each trial, the probability that the adversary can succeed is at most 2^{-40} (by (4.9)).

  Note that, with probability 2^{-40}, the adversary only obtains a correct system of non-linear equations. The system consists of $(k+1)^2+k+3$ unknowns and $(k+1)^2+k+3$ equations; i.e., 92 equations in 92 unknowns. It is not clear whether there is any efficient algorithm to solve the system.
- The adversary performs the man-in-the-middle attack as described in 4.1.2. The adversary needs a tag to be queried 16384 times to collect sufficient points (by (4.11)) to recover the tag's polynomial with probability 1, or it requires less queries but succeeds in each trial with a successful probability 2^{-42} (by (4.10)).
- The adversary tries to compute a valid response using messages previously generated by a tag. In each trial, the probability that the adversary can succeed is 2^{-63} (by (4.12)).
- The adversary answers a query with random responses. In each interaction with the reader, the probability that the adversary is identified as a valid tag is at most 2^{-22} when $N \leq 2^{32}$ (by (4.1)). On average, the adversary will succeed once if it tries this attack 2^{22} times. However, the adversary can only launch this attack when it is queried by a legitimate reader.
- The adversary tries to determine if two responses are from the same tag. For each pair of responses from one tag, the probability that the adversary succeeds is at most 2^{-63} (by 4.13).

- The adversary launches a Denial of Service (DOS) attack against a tag by repeatedly querying a tag. After being queried for 13700 times, a tag cannot be accepted by an authentic reader. In addition, an adversary may be able to trace the tag using information in addition to the tag's response, e.g., if the tag is accepted by an authentic reader.

Tag

In each session, a tag needs to generate $2b-1$ random numbers in F_{2^l}, evaluate a polynomial of degree k over F_{2^l}, and compute a hash function. Random number generation and hash computation have been used in most previous symmetric-key based RFID protocols (e.g., [82], [56], [35]). The tag needs to store m polynomials over F_{2^l}, each of degree k. So it needs to store $m(k+1)$ items of size l. For $m = 16, k = 8, l = 64$, it takes a 9216-bit ROM, corresponding to 9216 gates in hardware. Using Horner's rule, it takes k modular multiplication over F_{2^l} to evaluate a polynomial of degree k. A 64-bit modular multiplier takes several hundred gates in hardware. Therefore, in our protocol, a tag needs about 10000 more gates in hardware than a regular tag capable of hashing computation. As a comparison, a basic RFID tag with low security require a couple of thousands of gates [61], a hash function or a block cipher requires several thousands of gates [17], and about 20000 gates are required to implement an ECC processor for RFID tags to perform public key computations [47].

Server

The server stores m bivariate polynomials of degree k. This requires $m(k+1)^2 l/8 \approx 10$K bytes of memory.

In each session, for each bivariate polynomial $f(x,y)$ and each received point (z', r'), the reader needs to solve an equation $z' = f(x, h(r||r'))$. Then it checks if the roots are valid meta IDs. There are efficient algorithms to solve polynomials over finite fields, e.g., Berlekamp's algorithm [12] and the Cantor-Zassenhaus algorithm [23]. After a meta ID is computed, it takes constant time to check if it is valid. Solving the polynomials dominates the total time in an authentication process.

We implemented the reader algorithm based on NTL [1]. On a P4 3.2G PC with 1G RAM running Linux, when $m = 16, k = 8$, $b = 8$, and $l = 64$, the running time using the Cantor-Zassenhaus algorithm in average case (solving $mb/2$ polynomials of degree k on F_{2^l} where $l = 64$) is about 0.1 seconds for one query.

Scalability

N (the maximum number of tags) is at most l bits. N is also limited by the false positive probability given in (4.1). Given the fixed parameters $m = 16, k = 8$, $b = 8$, and $l = 64$, when the false positive probability is less than 2^{-22}, N can be as large as 2^{32}, which is sufficient for almost all conceivable applications. Therefore the protocol is highly scalable.

Comparison

We compare our protocol with OSK/AO [5]. Both protocols are secure and untraceable, and both are designed to solve the key search problem to provide good scalability.

Time and Space. OSK/AO has a query threshold Q_m, which is the length of its hash chain. OSK/AO needs to store a precomputed table of size M. The time for each query follows the rule

$$T = \mu \frac{(NQ_m)^2}{M^2}$$

where μ is a constant. For $N = 2^{20}$, $Q_m = 128$, and $M = 1$GByte, OSK/AO takes 0.004 milliseconds for one query. However, when the number of tags increases, without increasing the table size, the time increases fast. Let M, Q_m be fixed, and let T_1, T_2 be the time for tag number N_1, N_2 respectively. It holds that

$$T_2 = \left(\frac{N_2}{N_1}\right)^2 T_1.$$

If we consider a system of $N = 2^{32}$ tags, then, OSK/AO will take 67 seconds for one query, using an 1G bytes precomputed table. As a comparison, our protocol takes 0.1 seconds and 10K bytes memory in the server.

Query Threshold. In OSK/AO, if a tag is queried more than $Q_m = 128$ times by an adversary between two queries by an authentic server, then the tag cannot be recognized by an authentic reader. In our protocol, if a tag is queried a total of $Q_m = 13700$ times, then it cannot be recognized by an authentic reader. In OSK/AO, the time for one query increases in ${Q_m}^3$ (instead of ${Q_m}^2$). In our protocol, the time increases in $\sqrt{Q_m}$. Therefore, the query threshold Q_m in our protocol is much higher. However, OSK/AO does not have a limit on the total number of queries for a tag, although it is more susceptible to tag disable attacks where an adversary repeatedly queries a tag in a short time. On the contrary, our scheme is more resilient to tag disable attacks, but there is a limit on the total number of times that a tag can be queried. Which approach is better depends on how frequently a tag is queried. For example, in a library RFID system where a tag may be queried several times a day, the

OSK/AO scheme may be desirable. In some other applications such as e-passport where a tag is queried every several weeks or months, our protocol may be preferable.

Tag Hardware. The main drawback of our protocol compared to OSK/AO may be the hardware cost on the tag side, which is about 10000 more gates in hardware. As a comparison, it can be estimated that in an OSK/AO protocol, a tag may need 4000 gates, of which 2000 are for basic RFID functions and 2000 are for a hash function.

4.2 Rabin Encryption based Protocol

4.2.1 Security of Paddings

Hereinafter, we use $|x|$ to denote the length of a bit string x, and we use $x||y$ to denote the concatenation of strings x and y.

Integer Factoring Problem. Let $N = pq$ where p and q are large primes and $|p| \approx |q|$. The *factoring assumption* says that given N, for any polynomial time (in $|N|$) algorithm A and any polynomial Q, for sufficiently large $|N|$, it holds that $\Pr[(p,q) = A(N)] < 1/Q(|N|)$; i.e., it is infeasible to factor N in polynomial time (in $|N|$) with nonnegligible probability. $|N| = 1024$ is often chosen in practice.

Rabin Function. Let (N, p, q) be the parameters in the factoring assumption, and in addition, $p \equiv 3 \mod 4$ and $q \equiv 3 \mod 4$. The Rabin function computes $y = x^2 \mod N, x \in \mathbb{Z}_N^*$. The Rabin function is a trap-door one-way function in that given $y = x^2 \mod N$, without (p, q), to find x such that $y = x^2 \mod N$ is as hard as factoring N. With (p, q), x can be computed in polynomial time. Note that there are four distinct x values such that $x^2 = y \mod N$: $\pm x$ and $\pm \alpha x$ where α is a nontrivial square root of $1 \mod N$ (i.e., $\alpha^2 = 1 \mod N$ and $\alpha \neq \pm 1 \mod N$). Given x and αx, one can factor N [102, §5.8].

Randomized Rabin Function. Shamir [94] and Naccache [79] simultaneously proposed a randomized Rabin function. Instead of computing $y = x^2 \mod N$, the randomized Rabin function computes $y = x^2 + rN$ where r is a random number. The randomized Rabin function is as secure as the Rabin function when $|r|$ is big enough ($|r| \geq |N| + 80$ is recommended). The ways to compute $y = x^2 + rN$ in [94] and [79] are different. In [94], the numbers x, r and N are represented using a regular number system and the multiplication is conventional multiplication. In [79], the numbers are represented in a *residue number system* (RNS) and the multiplication is RNS multiplication. An RNS has a list of coprime numbers $p_1, \ldots, p_m$. Each number $x \leq \prod_{i=1}^m p_i$ is represented as a list of m numbers $x_i = x \mod p_i$. To compute

$z = x + y$ in RNS, one computes $z_i = x_i + y_i \mod p_i$. To compute $z = xy$ in RNS, one computes $z_i = x_i y_i \mod p_i$. Note that the resulting z should be in the range $0 \le z \le \prod_{i=1}^{m} p_i$. Given the RNS representation $(x_1, \ldots, x_m)$, x can be computed using the Chinese Remainder Theorem [102, §5.2.2].

The randomized Rabin encryption is as secure as the Rabin encryption if $|r| - |N|$ is sufficiently large. Usually it is recommended that r is 80 bits longer than N [94].

Short Padding Attacks. To use the Rabin function to encrypt a message m where $|m| < |N|$, some kind of padding is necessary. Some padding schemes are vulnerable to attacks based on the following result by Coppersmith [28]:

Theorem 4.2.1 (Coppersmith Theorem)**.** *Let N be an integer and let $f(x) \in \mathbb{Z}_N[x]$ be a monic polynomial of degree d. Then there is an efficient algorithm (denoted as Coppersmith algorithm) to find all $x_0 \in \mathbb{Z}$ such that $f(x_0) = 0 \mod N$ and $-N^{1/d} < x_0 < N^{1/d}$.*

In [28], several attacks are identified for the RSA function with short padding and a small encryption key (i.e., $y = x^3 \mod N$). The same attacks apply to the Rabin function as well (i.e., $y = x^2 \mod N$). We list these attacks, and rewrite them for the Rabin function as follows.

1. The padding function is $x = c||m$ where c is known to the adversary and $|m| < |N|/2$. Given the ciphertext $y = (c||m)^2 \mod N$, m can be computed as follows:

   Let $C = 2^{|m|}c$. Then it holds that

   $$y = m^2 + 2Cm + C^2 \mod N. \tag{4.14}$$

   (4.14) is a univariate polynomial in m of degree 2 (mod N). Since $|m| < |N|/2$, m can be computed using the Coppersmith algorithm.

2. The padding function is $x = r||m$ where r is a random string and $|r| < |N|/4$. Given two ciphertexts $y_1 = (r_1||m)^2 \mod N$ and $y_2 = (r_2||m)^2 \mod N$ for the same plaintext m, m can be computed as follows:

   Let $R_1 = 2^{|m|}r_1, R_2 = 2^{|m|}r_2, x_1 = m + R_1, \Delta = R_2 - R_1$. Then we have

   $$\begin{aligned} {x_1}^2 - y_1 &= 0 \mod N & (4.15)\\ (x_1 + \Delta)^2 - y_2 &= 0 \mod N & (4.16) \end{aligned}$$

   where x_1 and Δ are unknowns.

x_1 can be eliminated from (4.15) and (4.16) by taking their resultant[2], and it is easy to verify that the resultant equals to 0:

$$\begin{aligned} & \rho({x_1}^2 - y_1, (x_1+\Delta)^2 - y_2) \\ = \; & \Delta^4 - 2\Delta^2 y_2 - 2y_1\Delta^2 + {y_1}^2 - 2y_1y_2 + {y_2}^2 \\ = \; & 0 \mod N. \end{aligned} \tag{4.17}$$

(4.17) is a univariate polynomial in Δ of degree 4 (mod N). Since $|\Delta| < |N|/4$, Δ can be computed using the Coppersmith algorithm.

From (4.15) and (4.16) we have

$$\begin{aligned} (x_1+\Delta)^2 - {x_1}^2 &= 2x_1\Delta + \Delta^2 \\ &= y_2 - y_1 \mod N, \end{aligned}$$

then we can solve for x_1 and compute m by parsing $x_1 = r_1||m$.

SAEP Padding. In [19], Boneh proposed Simple OAEP (SAEP), a padding scheme for Rabin function which is provably secure in the sense that breaking the Rabin-SAEP scheme leads to factoring N. The SAEP padding is as follows: $x = m||o \oplus h(r)||r$ where r is a random string, $|r| > |N|/2$, o is a string of 0's, $|m| < |o|$, and h is a hash function. The Rabin-SAEP encryption provides semantic security under chosen ciphertext attacks, which implies semantic security under chosen plaintext attacks, meaning that even if the adversary can choose the plaintext m, he is not able to distinguish the ciphertext from a random string of the same size. The security proof is in the random oracle model which assumes h to be a random oracle. The proof is based on the Coppersmith Theorem.

The connection between our work and the above related work is as follows. WIPR is closely related to Shamir's randomized Rabin function using a simple padding, which may be vulnerable to the short padding attacks. One of our improvements to WIPR is to use RNS computation, which is similar to Naccache's randomized Rabin function. The other improvement is to use a secure padding similar to SAEP.

4.2.2 WIPR

WIPR was proposed in [85].

[2]The resultant of two monic polynomials P and Q is defined as

$$\rho(P,Q) = \prod_{(x,y):P(x)=0,Q(y)=0} (x-y).$$

Description

Setup. Let (p, q, N) be the parameters of the Rabin function and let $|N| = 1024$. In the scheme, RFID tags are provided with the public key N and a reader is provided with the secret key (p, q). Each tag is assigned with an ID.

Challenge. Reader generates a random bit string c where $|c| = 128$, and sends c to the tag.

Response. The tag generates a random bit string r, and computes $x = MIX(c||r||ID)$ where MIX is a simple byte-interleaving operation. The tag generates a random number r' where $|r'| = 1024 + 80$, then computes $y = x^2 + r'N$, and sends y to the reader.

Verify. The reader solves $x^2 \mod N = y \mod N$ for x. There are four roots. The reader checks if one of the roots contains c. If such a root is found, then the reader parses the root and finds ID.

WIPR is designed to provide the following properties:

- Secrecy. An adversary observing a protocol exchange between a reader and a tag cannot learn anything about the ID of the tag.

- Full backward and forward privacy. An adversary cannot determine whether a tag was a part of any past or future protocol exchange it has recorded, even if the adversary knows the ID of the tag.

The framework of WIPR is the same as Shamir's randomized Rabin function in [94]. The main novelty in WIPR is to use a reversible stream cipher to generate a long pseudorandom string on the fly which can be accessed forward and backward. The design of the reversible stream cipher is as follows. Let s_i be the i^{th} state of the stream cipher. To transfer to the next state, the cipher computes $s_{i+1} = s_{i-1} \oplus f(s_i)$ where f is a one-way function. The cipher can also transfer to the previous state by computing $s_{i-1} = s_{i+1} \oplus f(s_i)$. WIPR uses a boolean function to implement the oneway function f. The authors of WIPR noted that the boolean function is somewhat insecure.

WIPR did not specify some details of the scheme such as the MIX function and the sizes of ID and r. The authors of WIPR noted that the parameter sizes need to be fine-tuned, based on the relative strengths of attacks against the scheme's various subcomponents.

Analysis

Short Padding Attacks Against a Reduced WIPR WIPR uses a padding scheme $x = MIX(c||r||ID)$ before computing $y = x^2 \mod N$.[3] We consider a reduced version of WIPR where the MIX function is not used (i.e., $x = (c||r||ID)$). We discuss how the length of r affects the security of the protocol.

First, we show that, if $|r| < |N|/4 = 256$, then an adversary can compute the ID of a tag after querying the tag twice. The adversary queries the tag twice with the same challenge c, and receives $y_1 = (c||r_1||ID)^2 \mod N$ and $y_2 = (c||r_2||ID)^2 \mod N$. The number $x = c||r||ID$ can be expressed as $x = ac + br + ID$ where $a = 2^{|ID|+|r|}$ and $b = 2^{|ID|}$. Let $x_1 = ac + br_1 + ID$ and let $\Delta = (r_2 - r_1)$. Then the adversary has

$$\begin{aligned} {x_1}^2 &= y_1 \mod N \\ (x_1 + b\Delta)^2 &= y_2 \mod N \end{aligned}$$

and he can solve for Δ and x_1 as shown in case (2) of the short padding attacks in Section 4.2.1. Then ID can be found within x_1.

Next, we show that, when $|r| < |N|/2 = 512$, the scheme does not provide forward or backward privacy. Suppose the adversary gets a tag ID at some time. To tell if a message (c, y) observed at some other time involves this same ID, the adversary solves the equation

$$y = (ac + br + ID)^2 \mod N$$

for r using the Coppersmith algorithm. If (c, y) is generated using the ID; i.e., $y = (c||r||ID)^2 \mod N$ for some r where $|r| < \frac{|N|}{2}$, then according to the Coppersmith Theorem, the adversary can compute this r and verify that $y = (c||r||ID)^2 \mod N$. In this case, the adversary concludes that (c, y) is generated by this ID. In the other case where y is generated from another tag ID'; i.e., $y = (c||r'||ID')$ for some r', we show that there does not exist an r such that $(c||r||ID)^2 \mod N = y$. If such an r exists, then it can be computed using the Coppersmith algorithm. Now we get two square roots of y: $x_1 = c||r'||ID'$ and $x_2 = c||r||ID$. It is unlikely that $x_1 = -x_2$ since ID and ID' are independent, thus we can factor N using x_1 and x_2. Therefore, in this case, the Coppersmith algorithm will not output r such that $|r| < 512$ and $(c||r||ID)^2 \mod N = y$. Upon this result, the adversary can conclude that (c, y) is not related to the ID. Therefore, the adversary can successfully tell if the message (c, y) is related to a given ID. This attack is based on case (1) of the short padding attacks in Section 4.2.1.

[3]The tag actually computes $y = x^2 + rN$. It is equivalent to $y = x^2 \mod N$ in view of both functionality and security. In the analysis, we use the notation $y = x^2 \mod N$ for simplicity.

Parameter Choice for WIPR Two basic countermeasures can be taken in WIPR to withstand the short padding attacks.

- use a random padding with a length greater than 512 bits. In this case, even without a MIX function, the above short padding attacks do not work.
- use a MIX function to spread the random padding into at least three separated blocks, e.g., $MIX(c||r||ID) = (r_1||ID||r_2||c||r_3)$ where $r = r_1||r_2||r_3$. In this case, even when $|r| < 256$, the above short padding attacks do not work, either. Note that if r is separated into only two blocks, an attack is still possible, although the computation is more complicated and results are not guaranteed [28].

It is reasonable to assume that longer random paddings and more separated padding blocks are more resistant to the short padding attacks. In practice, c is 128 bits and we assume that ID is 128 bits, then r is 768 bits. A MIX function may divide r into 32 3-byte blocks, and insert one byte of $c||ID$ after each block (note that both the input and output of MIX are generated on the fly).

Reset Attacks In WIPR, a reversible stream cipher is designed to extend a short random string to a long pseudorandom string. Three such ciphers are used in WIPR. We note that the reversible stream cipher only serves to extend a short random seed to a long pseudorandom string for one identification session. One seed is needed for each individual session. Therefore, an additional random bit generator is needed for WIPR. We denote this random bit generator as RBG_1.

To ensure the security and privacy properties of WIPR, RBG_1 should be resistant to *reset attacks*, where the adversary can reset a tag's internal state to its initial state. Reset attacks have been considered for standard identification protocols in the smartcard setting [7], and they have recently been considered for RFID systems [16]. If RBG_1 is a pseudorandom bit generator (PRBG), and its output only depends on a secret initial seed, then the output of RBG_1 can be repeated after resetting. In this case, the adversary can recover the tag ID as follows. The adversary resets the tag, queries it with c_1, and gets the response $y_1 = (MIX(c_1||r||ID))^2 \mod N$. The adversary resets the tag again, queries it with c_2, and gets the response $y_2 = (MIX(c_2||r||ID))^2 \mod N$. The adversary chooses c_1 and c_2 such that they differ at only one bit. Suppose this bit is the i^{th} bit b_i in $x = MIX(c||r||ID) = x_1||b_i||x_2$, and $b_i = 1$ in c_1. Then the adversary can solve

$$\begin{aligned} y_1 &= (x_1||0||x_2 + 2^i)^2 \mod N \\ y_2 &= (x_1||0||x_2)^2 \mod N \end{aligned}$$

for $x_1||0||x_2$ and recover ID.

If RBG_1 is a true random bit generator, then the reset attack does not work. When RBG_1 is a PRBG, one way to avoid the above reset attack is to let the output of the RBG_1 depend on the challenge c. In this case, the above reset attack does not work, either.

We note that, if RBG_1 is a PRBG, then to ensure the forward and backward privacy of the protocol, the PRBG should also provide forward and backward security. More detailed discussion of these issues can be found in [110].

4.2.3 Improvements

When the padding length is long enough and the MIX function is properly designed, WIPR can be considered secure based on the fact that, despite considerable research interest over the last 30 years, there is still no way of performing Coppersmith-type attacks on ciphertexts in which c (known), r (random padding), and m (secret) are sufficiently mixed. However, as for any cryptographic protocol, "provable security" is usually preferable to security based on "no known attacks". Recall that, provable security means that, in a given security model, breaking the protocol leads to solving some hard problem, e.g., the integer factoring problem.

An essential building block of WIPR is the reversible stream cipher. The reversible stream cipher uses 1238 gates, which is highly lightweight and is critical for WIPR to be useful on RFID tags. However, as a cryptographic primitive, the reversible stream cipher has not received extensive public scrutiny, and hence, it may raise concerns about its security. It would be desirable to substitute it with a well-studied cipher. Also, we are interested in investigating possible ways to further reduce the hardware cost of WIPR without affecting its security.

Next, we propose two approaches to improve the security and to reduce the hardware cost of WIPR. The first is a secure padding based on SAEP. The resulting scheme is denoted as WIPR-SAEP. The major additional hardware cost for WIPR-SAEP is a hash function. WIPR-SAEP is provably secure in the sense that violating the security and privacy of the scheme leads to factoring N. The second improvement is to change the way to compute multiplication. We use RNS as in Naccache's randomized Rabin function [79]. The resulting scheme is denoted as WIPR-RNS. In WIPR-RNS, we replace the three reversible stream ciphers in WIPR with one regular stream cipher, hence we reduce the hardware cost and provide a better security guarantee. These two approaches can be used together, if desired.

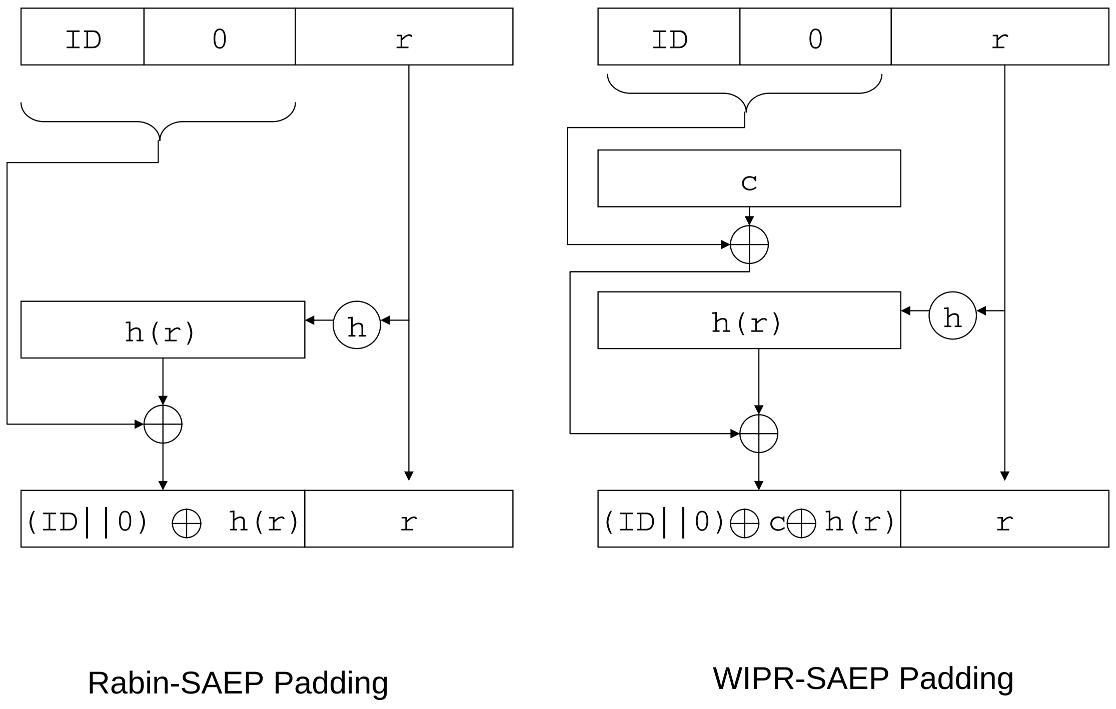


Figure 4.3: Rabin-SAEP and WIPR-SAEP Padding

WIPR-SAEP

Description We assume that the total number of tags is 2^s where $s < 64$. This would be sufficient for any conceivable application. The padding scheme is as follows:

$$x = (c \oplus (ID||o) \oplus h(r))||r$$

where $h()$ is a hash function, $|h()| = 128$, $|r| = 1024-128$, o is a string of 0, and $|o| = 128-s$. This padding is exactly the SAEP padding except that the additional parameter c has been included. Figure 4.3 illustrates the WIPR-SAEP padding and the Rabin-SAEP padding. The tag then computes $y = x^2 + r'N$ and sends y to the reader. The reader solves $x^2 \mod N = y \mod N$ for x, parses x as $x'||r$ where $|x'| = 128$, then computes $x'' = x' \oplus h(r) \oplus c$. If the low-order $128 - s$ bits in x'' are 0, then the high-order s bits comprise the ID.

As in WIPR, x needs to be generated on the fly and in two directions. In WIPR-SAEP, x is generated as follows. First, $h(r)$ is computed. Note that a hash function processes its input in multiple iterations, and the input is divided into contiguous fix-sized blocks. In each iteration, one block is read and processed. This allows r to be generated on the fly instead of being stored in RAM. Next, $(c \oplus (ID||o) \oplus h(r))$ is computed and stored in the RAM holding c. Then, x can be generated on the fly in two directions by regenerating r on the fly in two directions. Computation of $x^2 + r'N$ is the same as in WIPR. Compared to WIPR, WIPR-SAEP only needs one additional hash function.

Security Analysis We describe the following game to define the security and privacy of WIPR-SAEP. The notations r and o are the same as in the description for WIPR-SAEP. Since a randomized Rabin encryption is as secure as a conventional Rabin encryption, we will only write the encryption function as $y = x^2$.

WIPR-SAEP Game

- The adversary chooses two tags ID_0 and ID_1, and he sends (ID_0, ID_1) to the challenger. The challenger chooses a random bit b.
- This step repeats n times: The adversary chooses a random challenge c and sends c to the challenger. The challenger chooses a random r, computes $x = ((ID_b||o) \oplus c \oplus h(r))||r$, computes $y = x^2$, and sends y to the adversary.
- The adversary outputs b'. If $b' = b$, then the adversary wins.

It is clear that, if the adversary cannot win the game, then he cannot track a tag even if he knows the tag ID. Also the adversary cannot recover the ID by querying a tag.

We give a (sketch) proof that, when the hash function h is modelled as a random oracle, if the adversary can win the game, then the challenger can factor N. The challenger returns the adversary's hash queries and identification queries. Without loss of generality, we assume the challenger chooses m_0 in the game, and computes $x_0 = ((m_0||c) \oplus h(r))||r$ and $y = {x_0}^2$. For each y, there exists another square root $x_1 \neq x_0$, where $x_1 = ((m_1||c) \oplus h(r'))||r'$ for some r' and $h(r')$ values. Because $(m_b||c)$ is masked by $h(r)$ or $h(r')$, if the adversary can have any information about m_b, it must have queried $h(r)$ or $h(r')$, and with 1/2 probability, $h(r')$ is queried. In this case, the challenger knows r', $h(r')$, and c. Only $|m_1|$ consecutive bits in x_1 are unknown to the challenger. Let $x_1 = u||v$ where u is the block of unknown bits and v is the block of known bits. Since $|u| = |m_1| < |N|/4$ in the protocol, the challenger can solve $y = (u||v)^2$ for u, and then recover x_1. With x_0 and x_1, the challenger can factor N.

We conclude that if the factoring problem is hard, then the adversary cannot win the WIPR-SAEP game. WIPR-SAEP is secure against tracking even when the tag ID is disclosed; i.e., it provides forward and backward privacy, and the adversary cannot recover an ID by querying a tag.

Hash Function Selection WIPR-SAEP requires a hash function. For RFID tags, block cipher based hash functions are better candidates than dedicated hash functions such as SHA1 and MD5 [42], [17]. One example of such a hash function is H-PRESENT-128, which provides 128 bit output and requires 2330 hardware gates [17]. Some other hash functions dedicated to highly constrained devices provide other options. SQUASH-128 [95] is a keyed hash function that takes as input a 64 bit key and a 64 bit message, and outputs a 32 bit message authentication code (MAC). It is expected that SQUASH-128 requires about half of the number of the gates required by GRAIN-128 [54], which requires 2133 gates. Although SQUASH-128 cannot be directly used in WIPR, very low cost 128-bit hash functions using a similar design may be possible. By using the H-PRESENT-128 hash function, WIPR-SAEP requires a total of $5705 + 2330 = 8035$ gates in hardware.

WIPR-RNS

WIPR-RNS aims to reduce the hardware cost of WIPR and to remove the 1238-gate reversible stream ciphers used WIPR, which we consider to be a non-standard cryptographic primitive. Also the authors of WIPR noted that the boolean function used to construct the reversible stream cipher is somewhat insecure. In [79], RNS is used in the randomized Rabin function. But the goal of [79] is to reduce the time complexity, and several measures are used at the cost of additional ROM space. Here we use RNS to reduce the hardware cost of WIPR, and substitute the three reversible stream ciphers in WIPR with one regular stream cipher.

Description First, we give a high level description of WIPR-RNS. Let $p_1, \ldots, p_m$ be m coprime integers such that $|\prod_{i=1}^{m} p_i| > 2048 + 80$. $p_1, \ldots, p_m$ form the basis of an RNS. A tag receives a challenge c, generates x. Then the tag computes and sends $y = x^2 + r'N$ in RNS as in Algorithm 8.

After receiving $y_1, \ldots, y_m$, the reader can recover y and proceed with the verification process.

Next, we describe in detail the data representation and computation in Algorithm 8.

A regular stream cipher is used to generate r and r' on the fly. Note that in Algorithm

```
for i=1..m do
    $x_i = x \mod p_i$
    $r'_i = r' \mod p_i$
    $N_i = N \mod p_i$
    send $y_i = {x_i}^2 + r'_i N_i \mod p_i$
end
```

Algorithm 8: $x^2 + r'N$ in WIPR-RNS

8, r and r' do not have to be reversible as in WIPR. Therefore, a regular stream cipher is sufficient for WIPR-RNS. We may choose Grain [55] as the selected stream cipher. Grain uses about 1300 gates and may be the most efficient stream cipher without known flaws.

For the p_i's, we choose the 127 largest 16-bit primes and a set of 16-bit integers {64526, 64541, 64829, 64843, 65463, 65477, 65509}. It can be verified that the integers in the set are coprime. Therefore, the seven selected integers and the 127 primes are coprime and form the basis of an RNS. In this RNS, we can express integers less than $2^{2048+93}$, which is sufficient for WIPR-RNS. After these 134 coprime numbers are ordered, the difference between any two consecutive numbers is less than 15. Therefore, the 134 16-bit numbers can be stored in $16 + 4 \times 133 = 548$ bits in ROM, which costs 548 gates.

The tag computes $x = MIX(c||r||ID)$ on the fly the same way as in WIPR, except that x is generated in one direction, from most significant bit to least significant bit. Therefore, the reversible stream cipher in WIPR used to generate r in two directions can be replaced with a regular stream cipher.

To compute $x_i = x \mod p_i$, x is generated on the fly as described above, and x_i is computed using plain modular reduction.

N is stored in ROM as in WIPR. It is loaded in RAM on the fly and N_i is computed using plain modular reduction.

$y_i = {x_i}^2 + r'_i N_i \mod p_i$ is computed using regular multiplication and plain modular reduction.

The above algorithm uses a 32-bit adder and subtractor and a 16-bit multiplier, while the result is 2048+80 bits in length.

Analysis We compare WIPR and WIPR-RNS which uses WIPR padding in hardware cost and computational efficiency. Since we have not implemented WIPR-RNS in hardware, we only give an approximate estimation.

In WIPR-RNS, we use one regular stream cipher to generate r and r', and remove the three reversible stream ciphers in WIPR. This saves about 2500 gates. WIPR-RNS needs an additional 548 gates to store the RNS basis. The computing units (adder, subtractor, multiplier, and RAM to hold temporary results) may need several hundred more gates than WIPR. We estimate that more than 1000 gates can be saved in WIPR-RNS. Therefore, the estimated hardware cost of WIPR-RNS is $5705 - 1000 \approx 4700$ gates.

For time complexity, we compare the number of bit add/substract operations in computing $y = x^2 + r'N$. Let L be the length of x, N and r'. Let l be the length of p_i. To compute $x^2 + r'N$ as shown in Algorithm 8, the program runs $m = L/l$ rounds. In each round, it takes $3(L-1)l$ bit operations to compute x_i, r'_i, and N_i using plain modular reduction, and $3l^2 + 2l$ bit operations to compute ${x_i}^2 + r'_i N_i \mod p_i$. Therefore, the number of bit operations in WIPR-RNS is approximately

$$\frac{L}{l}(3(L-l)l + 3l^2) = 3L^2.$$

In WIPR, it is approximately $2L^2$. We estimate that WIPR is about 1.5 times faster than WIPR-RNS.

Combining SAEP and RNS

WIPR-SAEP and WIPR-RNS use two different approaches to improve the security of WIPR. WIPR-SAEP uses a secure padding, and WIPR-RNS replaces the non-standard stream cipher with a standard stream cipher. These two approaches can be combined together. The combined approach is the same as WIPR-RNS, except that it computes $x = (ID||o) \oplus c \oplus h(r)||r$ instead of $x = MIX(c||r||ID)$. x is generated the same way as described in WIPR-SAEP.

In Table 4.1, we give a summary of the security and the estimated hardware costs of WIPR-SAEP, WIPR-RNS, and WIPR-SAEP-RNS compared with WIPR. We assume that WIPR-SAEP uses a H-PRESENT-128 hash function.

	Number of Gates	Security
WIPR	5705	No proof
WIPR-SAEP	≈ 8000	Proof
WIPR-RNS	≈ 4700	No proof
WIPR-SAEP-RNS	≈ 7000	Proof

Table 4.1: Summary of hardware cost and security

4.3 Forward and Backward Privacy

Pseudorandom Bit Generator Random bit generation is very important in cryptography. However, true random bit generators rely on physically random processes, and hence they are inefficient or expensive in most practical environments. It is therefore more common to use *pseudorandom bit generators* (PRBG) in practice.

A standard (cryptographic) PRBG is a deterministic algorithm which, when given a truly random binary input of length n, outputs a binary sequence of length polynomial in n, say $p(n)$, which "appears" to be random. The input to the PRBG is called the seed, while the output of the PRBG is called a pseudorandom bit sequence. A standard PRBG is *secure*, if when given the first $l < p(n)$ bits of the output of the PRBG, it is infeasible in polynomial time (in n) to predict the next bit of the output ([76], [102]).

A *robust* PRBG provides additional security beyond a standard PRBG. In [6], Barak *et al.* propose a formal model and an architecture for robust PRBG, which satisfies the following properties of *forward security* and *backward security*[4]: (1) backward security: past output of the PRBG looks random to an adversary, even if the adversary learns the internal state at a later time. (2) forward security: future output of the PRBG looks random to an adversary with knowledge of the current state, provided that the PRBG is later refreshed with data of sufficient entropy. Similar properties and constructions can be found in [2] from NIST.

One example of standard PRBG is a block cipher with a secret key working in counter mode; i.e., if $E_k()$ is a block encryption scheme with secret key k, then the output is $s_i = E_k(i), i = 1, 2, \cdots$. Such a PRBG is a standard PRBG if the block cipher is secure, but it is not a robust PRBG.

Another example of standard PRBG is a keyed hash function with a secret key in output-feedback mode. Let $h_k()$ be a keyed hash function with secrete key k and an initial input s_0, the output is $s_{i+1} = h_k(s_i), i = 0, 1, \cdots$. Such a PRBG is not a robust PRBG, either.

In [98], the definition for a PRBG refers to the standard PRBG. It is worthwhile to point out that, in order to achieve forward untraceability and backward untraceability, the S-M scheme in [98] needs to use a PRBG stronger than the standard PRBG. In Section 4.3.1, we show that if a standard PRBG is used in the scheme, then the scheme may fail to provide forward untraceability and backward untraceability. We also construct an example of a robust PRBG for the S-M scheme to ensure its desired untraceability features.

[4]In [6], forward security means that past output is secure, while backward security means that future output is secure. Here we switch the two names to be consistent with the terminology we use in this paper.

4.3.1 Analysis Of The S-M Scheme

Scheme Description We briefly recall the S-M scheme which is described in [98]. $f_k()$ is a keyed hash function with key k. $h()$ is a hash function. $x \gg y$ denotes the operation that right rotate-shifts x by y bits and $x \ll y$ denotes the operation that left rotate-shifts x by y bits. l is the length of the parameters in the scheme.

Initially, for each tag T_i, the server stores its identifier (u_i, t_i). u_i is a unique secret for T_i and $t_i = h(u_i)$. The value t_i is stored in the tag. An identification session takes place as follows.

1. The reader sends a random challenge r_1 to the tag.

2. The tag generates a random r_2, computes $M_1 = t_i \oplus r_2, M_2 = f_{t_i}(r_1 \oplus r_2)$, and sends (M_1, M_2) to the reader.

3. The reader forwards (r_1, M_1, M_2) to the server.

4. The server searches for a tag T_i such that t_i satisfies $M_2 = f_{t_i}(r_1 \oplus r_2)$ where $r_2 = M_1 \oplus t_i$, computes $M_3 = u_i \oplus (r_2 \gg l/2)$, and sends M_3 to the reader. The server also updates u_i and t_i as follows: $u_{i(new)} = (u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2$, $t_{i(new)} = h(u_{i(new)})$.

5. The reader forwards M_3 to the tag.

6. The tag computes $u_i = M_3 \oplus (r_2 \gg l/2)$. If $h(u_i) = t_i$, then the tag updates t_i as follows: $t_i = h((u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r_2)$.

To perform the above protocol, a tag needs to generate random numbers. [98] cites a definition for PRBG which is just a standard PRBG.

Forward Untraceability We review forward untraceability as defined in [72] and [98]: at time τ, the adversary reveals the internal state of the tag T_i. At time $\tau' > \tau$, the tag performs a transaction with the server and the adversary does not eavesdrop on this transaction. Forward untraceability means that the adversary cannot tell if a transaction at time $\tau'' > \tau'$ involves the tag T_i.

We show that, if the PRBG used by the tag is not a robust PRBG, then the scheme does not achieve forward untraceability. To be concrete, we assume the block cipher based PRBG described in Section 4.3. Suppose at time τ, the adversary reveals the internal state of a tag, including its value t_i and the internal state of its PRBG. At some time $\tau' > \tau$, the tag has a transaction without being eavesdropped by the adversary. After time τ', the adversary observes multiple transaction messages.

Given the internal state (k and i) of the PRBG that the adversary obtained at time τ, the adversary can compute a sequence of n future outputs of the PRBG, where n is an upper bound on the maximum number of times that the tag could have invoked its PRBG since time τ. If any value r in this sequence and any observed message (r_1, M_1, M_2, M_3) after τ' satisfies $f_{M_1 \oplus r}(r_1 \oplus r) = M_2$, then the adversary can deduce that the message involves the tag T_i. In addition to identifying the tag, the adversary can further compute the t_i used in the observed transaction as well as the updated t_i after this transaction: $u_i = M_3 \oplus (r \gg l/2)$, current $t_i = h(u_i)$, and updated $t_i = h((u_i \ll l/4) \oplus (t_i \gg l/4) \oplus r_1 \oplus r)$. Therefore, the adversary can launch more attacks, e.g., to impersonate the tag or clone the tag.

Backward Untraceability We review backward untraceability as defined in [72] and [98]: at time τ, the adversary reveals the internal state of the tag T_i. Backward untraceability means that the adversary cannot tell if a transaction at time $\tau' < \tau$ involves the tag T_i.

Given the internal state (k and i) of the PRBG that the adversary obtained at time τ, the adversary can compute a sequence of i previous outputs of the PRBG. The adversary has also observed multiple transactions before time τ. If any r value in the computed output sequence and any observed message $(r_1, (M_1, M_2), M_3)$ satisfies $f_{M_1 \oplus r}(r_1 \oplus r) = M_2$, then the adversary can deduce that the message involves the tag T_i.

Using Robust PRBG In The S-M Scheme It is clear that, for the S-M scheme to achieve forward untraceability and backward untraceability, the PRBG used in the tags needs to be forward secure and backward secure. We can follow the construction in [6] or [2] to build a robust PRBG for the tags. An example of a robust PRBG is based on a block cipher working in counter mode. Let k be the secret key. Each time the PRBG is invoked, $s = E_k(i)$ is outputted as the random bits, the key is updated as $k = E_k(i + 1)$, and the counter i increases. The PRBG also refreshes its key as $k = k \oplus r_1$ each time a random challenge r_1 is received from the reader.

Now we briefly analyze the case when k and i are revealed at time τ. First we consider backward traceability. If E is a secure block cipher, then it is infeasible to find the previous keys it used; without the previous keys, it is infeasible to distinguish its previous output from a random number. Therefore backward untraceability of the schemes is preserved.

Next we consider forward traceability. Suppose in a certain transaction after τ, the adversary does not observe the messages, including r_1. Then the adversary does not know the updated key and thus he cannot predict the future outputs of E. Therefore, forward untraceability of the schemes is preserved.

Remark. We note that the backward security of the robust PRBG is sufficient but not necessary for the backward untraceability of the S-M scheme. Suppose the PRBG based on keyed hash function given in Section 4.3 is used in the S-M scheme. Given s_i and k at time τ, the adversary can tell if a given value x was generated by the PRBG by computing $x_1 = h_k(x_1), x_2 = h_k(x_2), \cdots$ and checking if s_i appears in the sequence. Therefore, this PRBG does not provide backward security. But given s_i and k at time τ, the adversary cannot compute any previous output of the PRBG. The attack described above for block cipher based PRBG does not work for the keyed hash based PRBG. However, we note that that the L-K scheme in [72] does need a robust PRBG to ensure its backward untraceability because the output of the PRBG is sent in plaintext.

Chapter 5

Conclusion and Future Work

5.1 Summary

In this thesis, cryptographic protocols, sensor network key management, and RFID authentication protocols were studied. The results are summarized as follows.

Cryptographic Protocols. We discussed the security model for deterministic two-move identification protocols. With the deterministic provers in such protocols, we simplified the CR2 model in [7] which captures concurrent attacks and reset attacks. Then we proposed an extremely simple identification protocol and proved that its CR2 security is equivalent to the hardness of the Strong Diffie-Hellman problem.

We proposed two AKE protocols that have efficient online computation and tight security proofs in the eCK model. Previous AKEs provide either efficient computation (e.g., MQV, HMQV, CMQV), or tight security proof (e.g., NAXOS, NETS), but not both. As an example, CMQV uses 2.17 exponentiations in computation, but does not have a tight security proof. NETS has a tight security proof, but it takes three exponentiations in computation. We prosed an AKE named SMEN whose online computation takes 1.25 exponentiations, close to that (1.17 exponentiations) of MQV, HMQV, and CMQV. The security reduction of SMEN is as tight as that of NAXOS in the eCK model. The NAXOS trick is used in the design of SMEN. We also proposed SMEN$^-$, which does not use the NAXOS trick. SMEN$^-$

takes 1.29 exponentiations in online computation. Without the NAXOS trick, SMEN$^-$ may be more resilient to static private key leakage. Both SMEN and SMEN$^-$ achieve efficient online computation and tight security reduction at the cost of one more exponentiation in offline computation and a longer message, and the limitation that one party is not allowed to establish a key with itself.

We showed that ElGamal encryption is OW-CCA1 under the strong generalized knowledge-of-exponent assumption (SGKEA) and the delayed-target discrete log assumption (DTDLA), and its security is equivalent to the hardness of the delayed-target computational Diffie-Hellman (DTCDH) problem. For DEG, we gave a simple proof that DEG is IND-CCA1 secure under the delayed-target decisional Diffie-Hellman assumption. We proposed a decisional DHK1 assumption (DDHK1), and proved that DHK1 implies DDHK1 and DEG is IND-CCA1 secure under the DDHK1 and DDH assumptions.

Sensor Network Key Management. We studied κ-edge-connectivity for two types of KPS and the resulting DSNs. We proved the κ value for a deterministic TD-type KPS, and we estimated the κ value for a random KPS, and validated the result using simulation. We also estimated and validated the κ values for DSNs constructed from TD-type KPSs and random KPSs. This approach may help to analyze the expected performance of other types of KPS and the DSNs constructed from them.

We also used simulations to validate the assumption that the block graph of a random KPS and its intersection with a random geometric graph can be modelled as a random graph, in that they are κ-edge-connected whenever the minimum node degree is κ. It would be desirable to prove the connections between the minimum node degree and κ-edge-connectivity of these graphs.

We proposed an enhanced security model to capture attacks against multi-path key establishment schemes in sensor networks. We identified two security objectives, which we term reliability and secrecy, that should be achieved. We observed that these objectives could be realized using perfectly secure message transmission schemes.

We proposed a new, optimal one-round PSMT scheme using Reed-Solomon codes, and we constructed two new multi-path key establishment schemes based on it. Both MPKE schemes achieve the desired objectives in an efficient manner. The second protocol potentially reduces the communication complexity in some cases by using feedback involving a message authentication code. Both of these protocols assume that the number of adversary-controlled paths is less than a 1/3 fraction of the number of paths connecting A and B.

We described another MPKE scheme that tolerates a higher fraction (less than 1/2) of paths controlled by the adversary. This scheme is based on a new protocol for a weakened version of message transmission, which is very simple and efficient.

RFID Authentication Protocols. We proposed a novel RFID protocol to solve the key search problem in RFID identification protocols. In previous RFID protocols, a hash-chain is used to achieve good privacy. In such protocols, to identify a tag, a server needs to search a table of size NQ, where N is the number of tags and Q is the length of the hash chain. The search takes either $\Theta(NQ)$ time or $\Theta(NQ)$ memory, and therefore it does not scale well. A time-memory tradeoff technique can mitigate the scalability problem. However, with the time-memory tradeoff, either the time or the space is still at least $\Theta((NQ)^{2/3})$. In our protocol, the server "solves", instead of "searches", for a tag ID. The protocol is based on polynomial operation, and its security and privacy is based on the difficulty of reconstructing a polynomial with noisy data. The protocol supports very large NQ values. In our demo implementation where $N = 2^{32}, Q = 13700$, the server takes 0.1 seconds and 10K bytes memory to identify a tag. As a comparison, a hash-chain protocol enhanced with time-memory tradeoff will take 67 seconds with the support of a 1G byte pre-computed table. At the same time, our protocol preserves security and privacy.

We analyzed the security and privacy of the WIPR RFID authentication protocol and proposed two approaches to improve its security and to further reduce its hardware cost. We showed that a reduced version of WIPR is vulnerable to short padding attacks, and WIPR is vulnerable to reset attacks if its PRBG is not properly chosen. We discussed countermeasures to withstand these attacks by properly specifying some details of WIPR. Then we proposed two variants of WIPR, namely, WIPR-SAEP and WIPR-RNS. WIPR-SAEP used SAEP padding to achieve provable security and privacy. WIPR-RNS used RNS computing to reduce the hardware costs of WIPR, and replaced the non-standard reversible stream cipher in WIPR with a standard stream cipher. The two approaches, SAEP padding and RNS computing, can be used together.

We analyzed an RFID identification scheme which is designed to provide forward untraceability and backward untraceability. We showed that, if a standard pseudorandom bit generator (PRBG) is used in the scheme, then the scheme may fail to provide forward untraceability and backward untraceability. To achieve these untraceability features, the scheme can use a robust PRBG which provides forward security and backward security.

5.2 Future Work

Authenticated Key Exchange. The SMEN and SMEN$^-$ AKE protocols achieve computational efficiency close to that of MQV, but they have some limitations. First, they should use a group G in which it is efficient to verify if $x \in G$ for a given x. A prime order elliptic curve group is such a group. To check if $x \in G$, it would suffice to verify that x is a point on the elliptic curve and that x is not the point at infinity. However, if G is a subgroup of $\mathbb{Z}_p^*$ where p is a large prime and G has a prime order q, then, to check if $x \in G$, we need to check if $x^q = 1$. In this case, SMEN and SMEN$^-$ need to compute one exponentiation more than MQV does in online computation to check if the received ephemeral public keys are valid, because there is one more ephemeral public key in SMEN/SMEN$^-$ than in MQV. Second, SMEN and SMEN$^-$ do not allow a party to establish a key with itself. It would be desirable to remove these limitations without affecting the efficiency and secrecy of the protocols.

A previous open problem for AKE protocols is to find a protocol that is as efficient as MQV and as secure as NAXOS [105]. This problem is not completely solved yet. At the same time, the presentation of SMEN$^-$ suggests a more difficult but reasonable problem. SMEN$^-$ lowers the risk of leaking the static private key in NAXOS and it also has a security reduction tighter than NAXOS. Therefore, SMEN$^-$ can be considered as more secure than NAXOS. As a result, the open problem may be updated to find a protocol that is as efficient as MQV and as secure as SMEN$^-$.

Multipath Key Establishment For existing sensor network multipath key establishment protocols, it has been assumed that a sender will identify the paths between itself and a receiver before it starts a key establishment session. To find a path in a network is the task of a routing protocol. However, current secure routing protocols for sensor networks require that the sender and the receiver share a secret key, or one of them has a public key. Therefore, under key predistribution, existing secure routing protocols cannot be used to identify the paths between a sender and a receiver that do not share a pre-assigned key. There is a deadlock between current solutions to path key establishment and secure routing in sensor networks: the solution to one problem relies on the solution to the other. It would be worthwhile to investigate how to break the deadlock; i.e., to find a solution to one of the problems without relying on the solution to the other.

Bibliography

[1] NTL: A library for doing number theory, `http://www.shoup.net/ntl/`.

[2] NIST special publication 800-90. Recommendation for random number generation using deterministic random bit generators. 2007.

[3] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.

[4] G. Avoine, E. Dysli, and P. Oechslin. Reducing time complexity in RFID systems. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 291–306. Springer, 2005.

[5] G. Avoine and P. Oechslin. A scalable and provably secure hash-based RFID protocol. In *PerCom Workshops*, pages 110–114. IEEE Computer Society, 2005.

[6] B. Barak and S. Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 203–212, New York, NY, USA, 2005. ACM.

[7] M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali. Identification protocols secure against reset attacks. In *EUROCRYPT 2001 Proceedings*, volume 2045 of *Lecture Notes in Computer Science*, pages 495–511, Berlin, Heidelberg, New York, 2001. Springer.

[8] M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali. Identification protocols secure against reset attacks. Unpublished manuscript, available from `http://www-cse.ucsd.edu/users/mihir/papers/id-reset.html` (this is the full version of [7]), 2001.

[9] M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In P.J. Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2004.

[10] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO '93 Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Berlin, Heidelberg, New York, 1994. Springer.

[11] C. H. Bennett, G. Brassard, and J.-M. Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17:210–229, 1988.

[12] E.R. Berlekamp. Factoring polynomials over finite fields. *Bell Systems Technical Journal*, (46):1853–1859, 1967.

[13] C. Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *MobiHoc '02: Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 80–91, New York, NY, USA, 2002. ACM Press.

[14] S.R. Blackburn and S. Gerke. Connectivity of the uniform random intersection graph. *Discrete Mathematics.* to appear.

[15] D. Bleichenbacher and P. Q. Nguyen. Noisy polynomial interpolation and noisy Chinese remaindering. In *EUROCRYPT*, Lecture Notes in Computer Science, pages 53–69. Springer, 2000.

[16] C. Blundo, G. Persiano, A.R. Sadeghi, and I. Visconti. Resettable and non-transferable chip authentication for ePassports. In *Conference on RFID Security*, Budaperst, Hongria, July 2008.

[17] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, and Y. Seurin. Hash functions and RFID tags: Mind the gap. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 283–299. Springer, 2008.

[18] B. Bollobás. *Random Graphs.* Cambridge University Press, second edition, 2001.

[19] D. Boneh. Simplified OAEP for the RSA and Rabin functions. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 275–291. Springer, 2001.

[20] Dan Boneh. The decision diffie-hellman problem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.

[21] Daniel R. L. Brown and Robert P. Gallant. The static diffie-hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004. `http://eprint.iacr.org/`.

[22] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 453–474, London, UK, 2001. Springer-Verlag.

[23] D. G. Cantor and H. Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Math. Comp.*, 36(154):587–592, 1981.

[24] S.A. Çamtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 15(2):346–358, 2007.

[25] D. Chakrabarti, S. Maitra, and B. Roy. A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design. *Int. J. Inf. Secur.*, 5(2):105–114, 2006.

[26] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor setworks. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 197, Washington, DC, USA, 2003. IEEE Computer Society.

[27] B. Chor, O. Goldreich, J. Hasted, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem or t-resilient functions. In *26th Annual Symposium on Foundations of Computer Science*, pages 396–407, 1984.

[28] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997.

[29] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Lecture Notes in Computer Science : Advances in Cryptology - EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 2000. Proceedings*, pages 392+, 2000.

[30] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 445–456, London, UK, 1992. Springer-Verlag.

[31] H.A. David. *Order Statistics.* John Wiley & Sons, Inc., second edition, 1981.

[32] J. Deng and Y.S. Han. Multipath key establishment for wireless sensor networks using just-enough redundancy transmission. *IEEE Transactions on Dependable and Secure Computing*, 5(3):177–190, 2008.

[33] R. Di Pietro, L.V. Mancini, A. Mei, A. Panconesi, and J. Radhakrishnan. Connectivity properties of secure wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 53–58, New York, NY, USA, 2004. ACM Press.

[34] R. Di Pietro, L.V. Mancini, A. Mei, A. Panconesi, and J. Radhakrishnan. How to design connected sensor networks that are provably secure. In *Proceedings of SecureComm 2006, the 2nd IEEE/CreateNet International Conference on Security and Privacy in Communication Networks*, 2006.

[35] T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 59–66, Washington, DC, USA, 2005. IEEE Computer Society.

[36] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40:17–47, 1993.

[37] W. Du, J. Deng, Y.S. Han, P. Varshney, J. Katz, and A. Khalili. A pairwise key pre-distribution scheme for wireless sensornetworks. *The ACM Transactions on Information and System Security (TISSEC)*, 8(2):228–258, May 2005.

[38] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[39] P. Erdős and A. Rényi. On the evolution of random graphs. In *Publ. Math. Inst. Hungar. Acad. Sci.*, volume 3, pages 17–61.

[40] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41–47, New York, NY, USA, 2002. ACM Press.

[41] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. *Journal of Cryptology*, 1:77–94, 1988.

[42] M. Feldhofer and C. Rechberger. A case against currently used hash functions in RFID protocols. In R. Meersman, Z. Tari, and P. Herrero, editors, *OTM Workshops (1)*, volume 4277 of *Lecture Notes in Computer Science*, pages 372–381. Springer, 2006.

[43] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO '86 Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Berlin, Heidelberg, New York, 1987. Springer.

[44] J.A. Fill, E.R. Scheinerman, and K.B. Singer-Cohen. Random intersection graphs when m= w(n): an equivalence theorem relating the evolution of the $g(n, m, p)$ and $g(n, p)$ models. *Random Struct. Algorithms*, 16(2):156–176, 2000.

[45] M. Fitzi, M. Franklin, J. Garay, and S. H. Vardhan. Towards optimal and efficient perfectly secure message transmission. In *TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 311–322. Springer, 2007.

[46] D. Freeman. Pairing-based identification schemes. Cryptology ePrint Archive, Report 2005/336, 2005. `http://eprint.iacr.org/`.

[47] F. Fürbass and J. Wolkerstorfer. ECC processor with low die size for RFID applications. In *ISCAS*, pages 1835–1838. IEEE, 2007.

[48] K. Gjøsteen. *Subgroup membership problems and public key cryptosystems.* PhD thesis, Norwegian University of Science and Technology, 2004.

[49] K. Gjøsteen. A new security proof for Damgård's ElGamal. In D. Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 150–158. Springer, 2006.

[50] E. Godehardt, J. Jaworski, and K. Rybarczyk. Random intersection graphs and classification. In *Advances in Data Analysis*, pages 67–74. Springer Berlin Heidelberg, 2007.

[51] L. Guillou and J.-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *CRYPTO '88 Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Berlin, Heidelberg, New York, 1990. Springer.

[52] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[53] D.R. Hare and W. McCuaig. The connectivity of the block-intersection graphs of designs. *Des. Codes Cryptography*, 3(1):5–8, 1993.

[54] M. Hell, T. Johansson, A. Maximov, and W. Meier. A stream cipher proposal: Grain-128. In *2006 IEEE International Symposium on Information Theory*, pages 1614–1618, 2006.

[55] M. Hell, T. Johansson, and W. Meier. Grain: a stream cipher for constrained environments. *Int. J. Wire. Mob. Comput.*, 2(1):86–93, 2007.

[56] D. Henrici and P. Müller. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In *PerCom Workshops*, pages 149–153. IEEE Computer Society, 2004.

[57] D. Huang and D. Medhi. A Byzantine resilient multi-path key establishment scheme and its robustness analysis for sensor networks. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 12*, page 240.2, Washington, DC, USA, 2005. IEEE Computer Society.

[58] W. C. Huffman and V. Pless. *Fundamentals of Error-Correcting Codes.* Cambridge University Press, 2003.

[59] J. Hwang and Y. Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 43–52, New York, NY, USA, 2004. ACM Press.

[60] A. Juels. Minimalist cryptography for low-cost RFID Tags. In Carlo Blundo and Stelvio Cimato, editors, *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164. Springer, 2004.

[61] A. Juels. RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, 2006.

[62] A. Kkiayias and M. Yung. Directions in polynomial reconstruction based cryptography. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer*, E87-A(5):978–985, 2004.

[63] A. Kkiayias and M. Yung. Cryptographic hardness based on the decoding of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 54(6), 2008.

[64] N. Koblitz and A. Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. *Journal of Mathematical Cryptology.*, 2(4):1862–2984, 2008.

[65] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In V. Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005.

[66] K. Kurosawa and K. Suzuki. Almost secure (1-round, n-channel) message transmission scheme. Cryptology ePrint Archive, Report 2007/076, 2007.

[67] K. Kurosawa and K. Suzuki. Truly efficient 2-round perfectly secure message transmission scheme. In *EUROCRYPT '08*, volume 4965 of *Lecture Notes in Computer Science*, pages 324–340. Springer, 2008.

[68] B.A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2007.

[69] L. Law, A.J. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28:119–134, 2003.

[70] J. Lee and D.R. Stinson. A combinatorial approach to key predistribution for distributed sensor networks. In *IEEE Wireless Communications and Networking Conference*, volume 2, pages 1200–1205, March 2005.

[71] J. Lee and D.R. Stinson. On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. *ACM Trans. Inf. Syst. Secur.*, 11(2):1–35, 2008.

[72] C.H. Lim and T. Kwon. Strong and robust RFID authentication enabling perfect ownership transfer. In P. Ning, S. Qing, and N. Li, editors, *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.

[73] D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, 2005.

[74] U.M. Maurer and S. Wolf. Diffie-Hellman oracles. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 268–282. Springer, 1996.

[75] R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Commun. ACM*, 24(9):583–584, 1981.

[76] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1996.

[77] D. Molnar and D. Wagner. Privacy and security in library RFID: issues, practices, and architectures. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 210–219, New York, NY, USA, 2004. ACM.

[78] D. M'Raïhi and D. Naccache. Batch exponentiation: a fast DLP-based signature generation strategy. In *CCS '96: Proceedings of the 3rd ACM conference on Computer and communications security*, pages 58–61, New York, NY, USA, 1996. ACM.

[79] D. Naccache, D. M'Raïhi, W. Wolfowicz, and A. di Porto. Are crypto-accelerators really inevitable? 20bit zero-knowledge in less than a second on simple 8-bit microcontrollers. In *EUROCRYPT'95*, pages 404–409, 1995.

[80] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254, New York, NY, USA, 1999. ACM.

[81] M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006.

[82] M. Ohkubo, K. Suzuki, and S. Kinoshita. Efficient hash-chain based RFID privacy protection scheme. In *In International Conference on Ubiquitous Computing Ubicomp, Workshop Privacy: Current Status and Future Directions*, 2004.

[83] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO '92 Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Berlin, Heidelberg, New York, 1993. Springer.

[84] T. Okamoto and D. Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2001.

[85] Y. Oren and M. Feldhofer. WIPR - a Public Key Implementation on Two Grains of Sand. In *Conference on RFID Security*, Budapest, Hungary, July 2008. `http://iss.oy.ne.ro/WIPR`.

[86] E.M. Palmer. *Graphical Evolution.* John Wiley & Sons, Inc., 1985.

[87] A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Unconditionally reliable and secure message transmission in undirected synchronous networks: possibility, feasibility and optimality. Cryptology ePrint Archive, Report 2008/141, 2008.

[88] M. Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.

[89] I.S. Reed and X. Chen. *Error-Control Coding for Data Networks*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[90] R. Rees, D. R. Stinson, R. Wei, and J. van Rees. An application of covering designs: Determining the maximum consistent set of shares in a threshold scheme. *Ars Combinatoria*, 53:225–237, 1999.

[91] H. Md. Sayeeda and H. Abu-Amara. Efficient perfectly secure message transmission in synchronous networks. *Information and Computation*, 126:53–61, 1996.

[92] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[93] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[94] A. Shamir. Memory efficient variants of public-key schemes for smart card applications. In *EUROCRYPT*, Lecture Notes in Computer Science, pages 445–449. Springer, 1994.

[95] A. Shamir. SQUASH — a new MAC with provable security properties for highly constrained devices such as RFID tags. In *Fast Software Encryption: 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, Lecture Notes in Computer Science, pages 144–157, Berlin, Heidelberg, 2008. Springer.

[96] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[97] K.B. Singer. *Random intersection graphs*. PhD thesis, The Johns Hopkins University, 1996.

[98] B. Song and C.J. Mitchell. RFID authentication protocol for low-cost tags. In *WiSec '08: Proceedings of the first ACM Conference on Wireless Network Security*, pages 140–147. ACM Press, 2008.

[99] D. R. Stinson and J. L. Massey. An infinite class of counterexamples to a conjecture concerning nonlinear resilient functions. *Journal of Cryptology*, 8:167–173, 1995.

[100] D. R. Stinson and S. Zhang. Algorithms for detecting cheaters in threshold schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 61:169–191, 2007.

[101] D.R. Stinson. *Combinatorial Designs*. Springer-Velag, 2003.

[102] D.R. Stinson. *Cryptography: Theory and Practice, Third Edition*. Chapman & Hall/CRC, Boca Raton, 2006.

[103] D.R. Stinson and J. Wu. An efficient and secure two-flow zero-knowledge identification protocol. *Journal of Mathematical Cryptology*, 1:201–220, 2007.

[104] P. Tague and R. Poovendran. A canonical seed assignment model for key predistribution in wireless sensor networks. *ACM Trans. Sen. Netw.*, 3(4):19, 2007.

[105] B. Ustaoğlu. *Key establishment - security models, protocols and usage*. PhD thesis, University of Waterloo, 2008.

[106] B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, 46(3):329–342, 2008.

[107] Y. Wang. Robust key establishment in sensor networks. *SIGMOD Record*, 33:14–19, 2004.

[108] R. Wei and J. Wu. Product construction of key distribution schemes for sensor networks. In *Lecture Notes in Computer Science*, volume 3357, pages 280–293. Springer Berlin / Heidelberg, 2004.

[109] H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932.

[110] J. Wu and D.R. Stinson. How to ensure forward and backward untraceability of RFID identification schemes by using a robust PRBG. Cryptology ePrint Archive, Report 2008/201, 2008. `http://eprint.iacr.org/`.

[111] J. Wu and D.R. Stinson. Minimum node degree and k-connectivity for key predistribution schemes and distributed sensor networks. In Virgil D. Gligor, Jean-Pierre Hubaux, and Radha Poovendran, editors, *WISEC*, pages 119–124. ACM, 2008.

[112] J. Wu and D.R. Stinson. An efficient identification protocol secure against concurrent-reset attacks, 2009. Manuscript.

[113] J. Wu and D.R. Stinson. A highly scalable RFID authentication protocol. In *Proceeding of the 14th Australasian Conference on Information Security and Privacy*, 2009. to appear.

[114] J. Wu and D.R. Stinson. How to improve security and reduce hardware demands of the WIPR RFID protocol. In *Proceeding of the 2009 IEEE International Conference on RFID*, 2009. to appear.

[115] J. Wu and D.R. Stinson. On the security of the ElGamal encryption scheme and Damgård's variant, 2009. Manuscript.

[116] J. Wu and B. Ustaoğlu. Efficient key exchange with tight security reduction, 2009. Manuscript.

[117] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In *ICNP*, pages 326–335. IEEE Computer Society, 2003.