Studies on Identity-Based Fault-Tolerant Key Distribution Systems

By Takeshi OKAMOTO

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Eiji OKAMOTO

February 15, 1999

Abstract

In 1984 A. Shamir [3] formulated the general idea of identity-based cryptosystem which is an asymmetric system employing users' identities instead of public keys, giving an example for ID-based signature system, and conceptual model for an ID-based encryption scheme. In this case, ID means information which is well-known to everyone. By using this system, we can solve the several problems which exist in public key cryptosystems such as management or procurement of the users' public keys.

In ID-based systems, there exit identity-based key distribution systems which are called ID-KDS for short. These systems have some advantages because they can be used not only for key distribution but also for authentication. In 1989, E.Okamoto and K.Tanaka [8] proposed a new ID-KDS which is based on the Diffie-Hellman key exchange scheme for key sharing, and which includes RSA-based authentication against impersonation. Besides this ID-KDS, there are many useful schemes [8] - [12].

These systems are efficient schemes for implementation, but they have certain drawbacks at the stage in which the center revoke a user's secret information and give a new one. That is, if a user's secret information is made public for some reasons, the center must discard the user's ID and adopt a different one. When it comes to implement ID-KDS, it is preferable that the center adopts one uniform ID such as a user's name, an e-mail address, a social security number, and so on. However, usual system is needed to make a file which contains several pieces of ID for one user. Therefore, these systems impose a burden on the users and lose the advantages of ID-based systems.

To solve these problems, we propose the following concept. That is, even after the center has revoked a user's secret information, the center generates a new one without any change of ID. This means that it keeps the one-to-one correspondence between users and ID's. Therefore, we must generate several pieces of secret information for a piece of ID. In this thesis, we realize this concept by modifying the Okamoto-Tanaka key exchange scheme [8].

In this thesis we study the following themes:

1. We propose a new concept of identity-based cryptosystem and call this system "Identity-based fault-tolerant key distribution system".

2. To realize above concept, we propose a concrete scheme by modifying the Okamoto-Tanaka key exchange scheme.

3. We study the security of the proposed scheme by using reduction of functions.

4. We consider the applications of the proposed concept to expand it into other key management.

Acknowledgments

First of all, I am most grateful to my supervisor, Professor Eiji Okamoto, who gave me the opportunity to study in cryptography and supported me. I am grateful to thank Professor Tetsuo Asano and Associate Professor Mineo Kaneko for their much advice and encouragement. I would like to thank Associate Professor Atsuko Miyaji for helpful comments and nice discussions with her.

I wish to express my gratitude to Associate Professor Masahiro Mambo at Tohoku University and Professor Masahiro Kubota at Kyoto Institute of Technology for their precious advice and helpful support.

I greatly thank Associate Mitsuru Tada and Associate Xun Yi for their eager suggestions and helpful comments. I deeply thank Dr.Hisao Sakazaki and Mr.Shigeki Kitazawa for many comments and helpful teaching. They guided me into the fascinating field of cryptography and the basis for this thesis lies in the joint work with them.

I really appreciate to take this opportunity and thank all the people who supported my research and gave me good suggestions. I would like to thank all the members in Okamoto-Miyaji Laboratory for their helpful comments, nice talks and much advice. In our laboratory, I always had a great and relaxing atmosphere.

Finally, I sincerely thank my parents, Hideo Okamoto, Tomiko Okamoto, my brothers, Tsuyoshi Okamoto, Satoshi Okamoto and my sister, Megumi Okamoto for their support, advice and encouragement.

Contents

Chapter 1

Introduction

1.1 Preface

Cryptography is a strategy of information protection that dates back four thousand years. It is an ancient art that is taken on new significance in today's information society.

Through the ages, cryptography has protected communications while they were being transmitted through hostile environments - usually involving war or diplomacy. Especially, cryptography in World War II owed its biggest boom to the scientific mobilization. The world's first digital computers were built to crack codes at that time.

In 1949 the publication by C. E. Shannon of the paper, "Communication Theory of Secret Systems", ushered in the era of scientific secret key cryptography. Shannon provided a theory of secrecy systems almost as comprehensive as the theory of communications.

In 1977 Data Encryption Standard (DES) was published by National Bureau of Standards. The whole idea of a "standard" in cryptography is certainly revolutionary. Before the publication of DES, there apparently were no publications containing a complete algorithm for practical cryptographic usage.

The real breakthrough of the cryptography came with the publication in 1976 by W.Diffie and M.E.Hellman of their work "New Directions in Cryptography" [1]. In this paper, they proposed the concept of public key cryptography and showed that secret communication is possible without an exchange of secret key in advance, while usual symmetric cryptosystem was required for such preparations. Their splendid idea was to use two different keys, a public key for encryption and a private key for decryption. Based on this asymmetry, they further proposed the concept of digital signatures. Here, the private key is used to sign a message and the public key is used to verify a signature. However, they did not provide realizations of the new concepts, but they proposed a protocol that allows two entities to share a common secret key only by exchanging information in public.

The concept of public key cryptography inspired many researchers, and it soon became a fast-growing and fascinating research theme. In the following years, although many realization of public key encryption and digital signature schemes were proposed, most notable one was RSA scheme. This scheme was introduced by three inventors R.L.Rivest, A.Shamir and L.Adleman who published the paper “A method for obtaining digital signatures and public key cryptosystems” [2] in 1978. This scheme was the first practical public-key encryption and digital signature schemes. Based on these primitives, more complex systems such as digital payment schemes or voting schemes were devised.

On the other hand, there are several problems in public key cryptosystems. That is, each user must have a file which contains users’ public keys, and if one user wants to send a message to another, procurement of users’ public keys is very costly.

To solve these problems, in 1984 A.Shamir [3] formulated the general idea of identity-based cryptosystem which is an asymmetric system employing users’ identities instead of public keys, giving an example for ID-based signature system, and conceptual model for an ID-based encryption scheme. In this case, ID means information which is well-known to everyone. In ID-based systems, there are identity-based key distribution systems which are called ID-KDS for short. These systems have some advantages because they can be used not only for key distribution but also for authentication. In 1989, E.Okamoto and K.Tanaka [8] proposed a new ID-KDS which is based on the Diffie-Hellman key exchange scheme for key sharing, and which includes RSA-based authentication against impersonation.

In these days as a remarkable characteristic of modern cryptography, cryptography has been used for network security. Especially Internet which is a sort of network system, has enabled us to communicate with each other on networks which reach around the world. However, it has caused some problems such as wiretapping, forgery and impersonation, which have been getting terribly serious. Since the progress of cryptosystem is necessary to realize a secure communication, it is preferable that communication systems give users less burden and more secure environment. These things can establish practical infrastructures for network communications.

To solve these problems, we can adopt the technique of ID-KDS. Regarding this system, many useful schemes [8] - [12] are proposed up to now. These systems are efficient schemes for implementation, but they have certain drawbacks at the stage in which the center revokes and renews a user’s secret information. That is, when the center revokes a user's secret information on the assumption that it is public for some reasons, the center must discard the user's ID and use the different one. To determine the ID information, it is preferable that the center adopts one uniform ID such as a user's name, an e-mail address, or a social security number. In these systems, the user need to make a file which contains several pieces of ID for one user. Therefore, these systems impose a burden on users and lose the advantages of ID-based systems.

The concept of our proposal is as follows: Even after the center has revoked a user's secret information, the center generates a new one without any change of ID. This means that it keeps the one-to-one correspondence between users and ID's. Therefore, we must generate several pieces of secret information for a piece of ID. In this paper, we realize this concept by modifying the Okamoto-Tanaka key exchange scheme [8].

1.2 Thesis Outline

Our thesis is organized as follows.

Chapter 2 summarizes the public-key cryptosystem and shows several famous encryption and signature schemes.

Chapter 3 examines several aspects of the key management. One aspect is the importance of the keys employed by secure algorithms and methods. Another aspect is authorized key management methods.

Chapter 4 shows the overview of Turing machine at first, and indicates mathematically precise definitions for complexity classes, reductions and functions to break several protocols. This chapter also shows the ordering among difficulty of functions and finally, indicates reductions among functions. Each theorem in this chapter was proved by M.Mambo and H.Shizuya [17].

Chapter 5 shows a new concept of identity-based cryptosystem and proposes a new identity-based key distribution system. Security considerations of our proposed scheme are studied by using reductions among functions. The conceptual structure of our proposed scheme is also discussed.

Chapter 2

Public-Key Cryptosystems

Public-key cryptography provides a radical departure from all that has gone before. For one thing, public-key algorithms are based on mathematical functions rather than on substitution and permutation. But more important, public-key cryptography is asymmetric, involving the use of two separate keys, in contrast to the symmetric conventional encryption, which uses only one key.

The idea behind a "*public-key*" system is that it might be possible to find a cryptosystem which it is computationally infeasible to determine the decryption rule D_K even by using the encryption rule E_K. If so, then the E_K could be made public by publishing it in a directory (hence we call this "*public-key*" system). the advantage of a public-key system is that Alice (or anyone else) can send an encrypted message to Bob (without the prior communication of a secret key) by using the public encryption rule E_K. Bob will be the only person that can decrypt the ciphertext, using his secret decryption rule D_K.

2.1 Applications of Public-Key Cryptosystems

In broad terms, we can classify the use of public-key cryptosystems into three categories:

1. *Encryption/Decryption* : The sender encrypts a message with the recipient's public key.

2. *Digital Signature* : The sender "*signs*" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is bound in some way to the message.

3. *Key exchange* : Two sides cooperate to exchange a session key. Several quite different approaches are possible, involving the private key(s) of one or both parties.

Some algorithms are suitable for all three applications, whereas others can only be used for one or two of these applications.

2.2 Requirement for Public-Key Cryptography

We show algorithms which public-key cryptography must fulfill:

1. It is computationally easy for Bob to generate a pair (PK_B, SK_B), where PK_B is a public key and SK_B is a secret key.
2. It is computationally easy for a sender Alice who know the public key PK_B and the message M which is encrypted by Alice to generate the corresponding ciphertext
$$C = E_{PK_B}(M).$$
3. It is computationally easy for the receiver Bob to decrypt the resulting ciphertext C using the private key to recover the original message
$$M = D_{SK_B}(C) = D_{SK_B}[E_{PK_B}(M)].$$
4. It is computationally infeasible for an opponent who know the public key PK_B to determinate the secret key SK_B.
5. It is computationally infeasible for an opponent who know the public key PK_B and a ciphertext C to recover the original message M.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications:

6. The encryption and decryption functions can be applied in either order
$$M = E_{PK_B}[D_{SK_B}(M)].$$

There are formidable requirements, as evidenced by the fact that only one of such algorithm has received widespread acceptance in over a quarter-century since the concept of public-key cryptography was proposed.

Before elaborating on why the requirements are so formidable, let us first recast them. The requirements boil down to the need for a trapdoor one-way function. "*one-way function*" is one that maps a domain into a range such that every function value is easy whereas the calculation of the inverse is infeasible:

$$\begin{aligned} Y &= f(X) && \text{easy.} \\ X &= f^{-1}(Y) && \text{infeasible.} \end{aligned}$$

Generally, "*easy*" is defined to mean a problem that can be solved in polynomial time as a function of input size. Thus, if the length of the input is n bits, then the time cost to compute the function is proportional to n^a, where a is a fixed constant. Next, the term "*infeasible*" is a much fuzzier concept. In general, we can say a problem is infeasible if the effort to solve it grows faster than polynomial time as a function of input size. For example, if the length of the input is n bits and the time to compute the function is proposition to 2^n, it is considered infeasible. Unfortunately, it is difficult to determine if a particular algorithm exhibits this complexity. Furthermore, traditional notions of computational complexity focus on the worst-case of average-case complexity of an algorithm. These measures are worthless for cryptography, which requires that it be infeasible to invert a function for virtually all inputs, not for the worst case or even average case.

We now turn to the definition of a "*trapdoor one way function*", which, like the one-way function, is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known. With the additional information, the inverse can be calculated in polynomial time. We can summarize as follows: A trapdoor one-way function is a family of invertible functions f_K, such that,

$$\begin{array}{lll} Y & = f_K(X) & \text{easy, if } K \text{ and } X \text{ are known.} \\ X & = f_K^{-1}(Y) & \text{easy, if } K \text{ and } Y \text{ are known.} \\ X & = f_K^{-1}(Y) & \text{infeasible, if } Y \text{ is known but } K \text{ is not known.} \end{array}$$

Thus, the development of a practical public-key scheme depends on discovery of a suitable trapdoor one-way function.

2.3 Encryption Schemes

We show two famous encryption schemes, such as the RSA and ElGamal encryption schemes. For preparation, we assume:

1. Alice generates several keys.
2. Bob encrypts a message m and makes a cipher text c for Alice.
3. Alice decrypts the cipher text c and get the message m.

Let h be a collision-resident hash function such as

$$h : \{0,1\}^* \mapsto \{0,1\}^{t-1},$$

where t is the security parameter.

2.3.1 RSA Encryption Scheme

Key generation

Alice picks two large primes p and q, and computes n, $\lambda(n)$, where $n = pq$ and $\lambda(n) = \operatorname{lcm}(p-1, q-1)$. Alice selects $e \in Z^*_{\lambda(n)}$ and computes d satisfies $ed = 1 \pmod{\lambda(n)}$. In this case, Alice's public key is (n, e) and secret key is (p, q, d).

Encryption

Bob computes

$$c = m^e \pmod{n}$$

and sends c to Alice.

Decryption

Alice computes

$$m = c^d \pmod{n}.$$

She can get the message m since

$$\begin{aligned} c^d &= (m^e)^d \\ &= m^{ed} \\ &= m \qquad \pmod{n}. \end{aligned}$$

2.3.2 ElGamal Encryption Scheme

Key generation

Alice picks a large prime p and a generator g of Z_p^*. Alice selects a random integer $x \in Z_{p-1}^*$ and computes $y = g^x \pmod{p}$. In this case, Alice's public key is (p, g, y) and secret key is x.

Encryption

Bob picks a random integer $k \in Z_{p-1}$ computes

$$\begin{aligned} c_1 &= g^k &\pmod{p}, \\ c_2 &= m \cdot y^k &\pmod{p}, \end{aligned}$$

and sends (c_1, c_2) to Alice.

Decryption

Alice computes

$$m = \frac{c_2}{c_1^x} \pmod{p}.$$

She gets the message m since

$$\begin{aligned} \frac{c_2}{c_1^x} &= \frac{m \cdot y^k}{(g^k)^x} \\ &= \frac{m \cdot (g^x)^k}{g^{kx}} \\ &= m \qquad \pmod{p}. \end{aligned}$$

2.4 Digital Signature Schemes

We show two famous signature schemes, such as the RSA and ElGamal Signature schemes. For preparation, we assume:

1. Alice generates several keys.
2. Alice signs a message m and makes a signature s.
3. Bob verifies the signature s.

Let h be a collision-resident hash function such as

$$h : \{0,1\}^* \mapsto \{0,1\}^{t-1},$$

where t is the security parameter.

2.4.1 RSA Signature Scheme

Key generation

Alice picks two large primes p and q, and computes n, $\lambda(n)$, where $n = pq$ and $\lambda(n) = \mathrm{lcm}\,(p-1, q-1)$. Alice selects $e \in Z^*_{\lambda(n)}$ and computes d satisfies $ed = 1 \pmod{\lambda(n)}$. Alice publishes (n, e) and keeps (p, q, d) secret. In this case, Alice's signature key is (p, q, d) and certification key is (n, e).

Signature generation

Alice computes

$$s = h(m)^d \pmod{n},$$

and sends (m, s) to Bob.

Signature verification

Bob confirms whether the following verification holds:

$$h(m) \stackrel{?}{=} s^e \pmod{n}.$$

If it is true, Bob accepts the signature. Otherwise, Bob rejects it.

2.4.2 ElGamal Signature Scheme

Key generation

Alice picks a large prime p, and a generator g of Z^*_p. Alice selects a random integer $x \in Z^*_{p-1}$ and computes $y = g^x \pmod{p}$. In this case, Alice's signature key is (p, g, y) and certification key is x.

Signature generation

Alice picks an integer $k \in Z_{p-1}^*$, computes

$$\begin{array}{lll} r & = g^k & \pmod{p}, \\ s & = k^{-1}(h(m) - xr) & \pmod{p-1}, \end{array}$$

and sends (m, r, s) to Bob.

Signature verification

Bob confirms whether the following verification holds:

$$g^{h(m)} \overset{?}{=} y^r r^s \pmod{n}.$$

If it is true, Bob accepts the signature. Otherwise, Bob rejects it.

Chapter 3

Key Establishment Protocols

3.1 General Aspects of Key Management

Since the security of many cryptographic algorithms and methods depends to a large extent on the secrecy of the key, the key must be kept secret, no matter how ingenious and safe the algorithm may be. Whoever has access to the key, can also access the information, assume someone else's identity, etc. This applies not only to symmetrical systems, which require unconditional secrecy of all keys, but also to asymmetrical systems, which are based on both public and secret keys.

These problems are considered in "*key management*". In general, a key management system must not only prevent intruders from obtaining a key, but in addition, it must avoid unauthorized use of keys, deliberate modification and other forms of manipulation of keys, etc. It is desirable to be able to detect any situation in which this occurs. Naturally, once the reliability of a key is impaired, its use must be terminated immediately.

3.2 Key Management Techniques

3.2.1 Problems with Symmetric Key Cryptography

We assume the system which n users involve symmetric-key techniques. If each pair of users may potentially need to communicate securely, then each pair must share a distinct secret key. In this case, each party must have $n-1$ secret keys; the overall number of keys in the system, which may need to be centrally backed up, is then $n(n-1)/2$, or approximately n^2. As the size of a system increases, this number becomes unacceptably large.

For systems based on symmetric-key techniques, the solution is to use centralized key servers with a trusted third party at the center or hub of communications. These methods diminish the n^2 key distribution problem, at the cost of the requirement of an on-line trusted server, and additional communications with it. Public-key techniques give an alternate solution.

3.2.2 Key Distribution Models

Point-to-point communications and "*centralized key management*", are examples of simple key distribution (communications) models. As an example of the centralized key management models, we indicate "*key distribution center*" model. These models are described below, where K_{XY} denotes a key shared by X and Y.

1. *point-to-point* mechanisms: These involve two parties communicating directly.
2. *key distribution centers* (KDCs): KDCs are used to distribute keys between users which share distinct keys with the KDC, but not with each other. A basic KDC protocols as follows. Upon request from Alice to share a key with Bob, the Center generates or otherwise acquires a key K, then sends it encrypted under K_{AC} to Alice, along with a copy of K (for Bob) encrypted under K_{BC}. Alternatively, Center may communicate K (secured under K_{BC}) to Bob directly.

Note that centralized key management involving third parties (KDCs in this case) offers the advantage of key-storage efficiency: each party need maintain only one long-term secret key with the trusted third party (rather than one for each potential communications partner). Potential disadvantage include: vulnerability to loss of overall system security if the central node is compromised (providing an attractive target to adversaries); a performance bottleneck if the central node becomes overloaded; loss of service if the central node fails (a critical reliability point); and the requirement of an on-line trusted server.

3.3 Point-to-Point Key Management

3.3.1 Key Distribution for Symmetric Algorithms

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized. Although full decentralization is not practical for larger networks, it may be useful with in a local context.

A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution. Therefore, there need to be as many as $n(n-1)/2$ master keys for a configuration with n end systems.

A session key may be established with the following sequence of steps;

1. Alice issues a request to be for a session key. The message includes the identity of Alice and Bob and a unique identifier N_1 for this transaction, which we refer to as a "*nonce*". The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differ with each request. Also, to prevent masquerade, it is desirable for it to be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.

2. Bob responds with a message that is encrypted using the shared master key. The response includes the session key selected by Bob, an identifier of Bob, the value $f(N_1)$, and another nonce, N_2.

3. Using the new session key, Alice returns $f(N_2)$ to Bob.

   Although each mode must maintain at most $(n-1)$ master keys, as many session keys as required may be generated and used. Since the message transferred using the master key are short, cryptanalysis is different. As before, session keys are used for only a limited time to protect them.

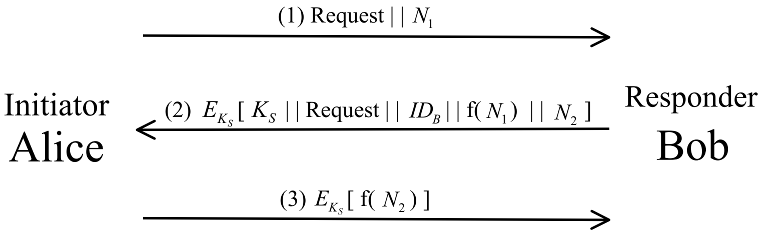

(1) Request | | N_1
Initiator
Alice
(2) E_{K_S}[K_S | | Request | | ID_B | | f(N_1) | | N_2]
Responder
Bob
(3) E_{K_S}[f(N_2)]

Figure 3.1: Point-to-point key management.

3.3.2 Diffie-Hellman Key Exchange Scheme and Its Attack

Diffie-Hellman key exchange scheme provide the first practical solution to the key distribution problem allowing two parties, never having met in advance or shared key material, to establish a shared secret key by exchanging messages over an open channel. Though this scheme have several kinds of models, we show the basic one.

Protocol

Preparation phase

An appropriate a large prime p, and a generator g of Z_p^* are select and published. Here, (p,g) serves as the users' public key.

Common key generation phase

Alice selects a random integer $a \in Z_{p-1}^*$,computes

$$y_A = g^a \pmod{p},$$

and sends y_A to Bob. Similarly, Bob selects a random integer $b \in Z^*_{p-1}$, computes

$$y_B = g^b \pmod{p},$$

and sends y_A to Alice. Next, Alice computes

$$K_A = y_B^a \pmod{p}.$$

Similarly Bob computes

$$K_B = y_A^b \pmod{p}.$$

K_A and K_B serve as a common key since

$$\begin{aligned} K_A &= y_A^b \\ &= (g^a)^b \\ &= g^{ab} \\ &= K_B \pmod{p} \end{aligned}$$

Attack Strategy

Unfortunately, the basic type of the Diffie-Hellman scheme is vulnerable to an active adversary who uses an "*intruder-in-the-middle*" attack. We show the example by using this attack.

We assume Alice and Bob have private keys a and b,respectively and Carol creates a' and b'. Carol intercepts Alice's exponential and replaces it by $g^{a'}$. Similarly, she intercepts Bob's exponential and replaces it by $g^{b'}$. Alice forms session key $K_A = g^{ab'}$, while Bob forms session key $K_B = g^{a'b}$. Carol is able to both these keys. When Alice subsequently sends a message to Bob encrypted under K_A, Carol deciphers it, re-enciphers under K_B, and forwards it to B. Similarly Carol deciphers message encrypted by Bob for Alice under K_B and re-enciphers them under K_A. Alice and Bob believe they communicate securely, while Carol reads all traffic.

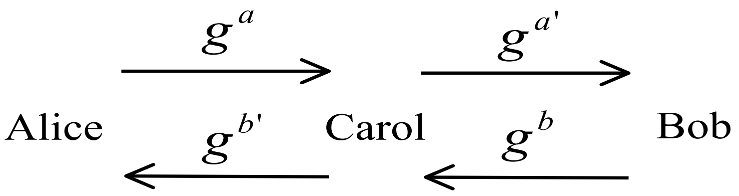


Figure 3.2: Intruder-in-the-middle attack.

To prevent this attack, it is essential for Alice and Bob to make sure that they are exchanging messages with each other and not with Carol. Before exchanging keys, Alice and Bob might carry out a separate protocol to establish each other's identity, for example by using one of the identification. But this offers no protection against an intruder-in the middle attack if Carol simply remains inactive until after Alice and Bob have proved their identities to each other. Hence, the key agreement protocol should itself authenticate the participants' identities at the same time as the key is being established. Such a protocol will be called "*authenticated key agreement*".

3.4 Centralized Key Management

3.4.1 Key Agreement for Symmetric Algorithm

We show a typical symmetric model illustrated in Figure 3.3. This model assumes that each user shares a unique master key with the key distribution center(KDC).

Let us assume that user Alice wishes to establish a logical connection with Bob and require a one-time session key to protect the data transmitted over connection. Alice has a secret key K_A known only for itself and the KDC; similarly, Bob shares the master key K_B with the KDC. The following steps occur:

1. Alice issues a request to the KDC for a session key to protect a logical connection to Bob. The message includes the identity both Alice and Bob, and a nonce N_1

2. The KDC responds with a message encrypted by using K_A. Therefore, Alice is the only one who can successfully receive the message, and Alice knows that it originated at the KDC. The message includes two items instead for Alice:

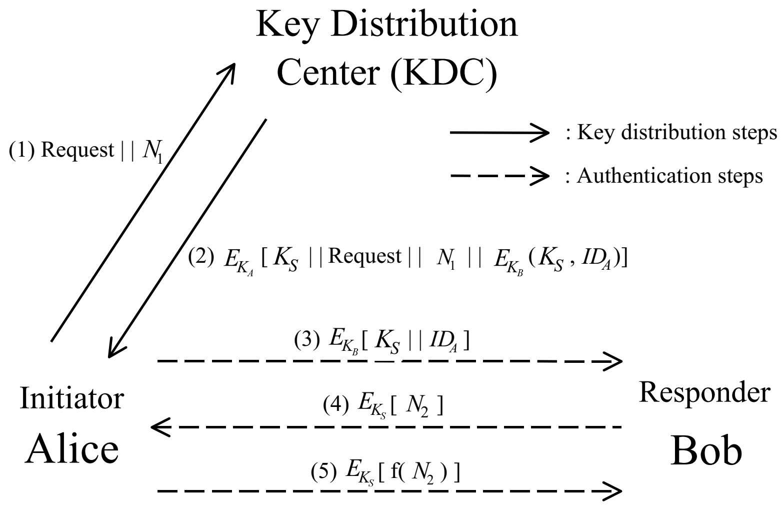
   - The one-time session key K_s which is used for the session;
   - The original request message, including the nonce, to enable Alice to match this response with the appropriate request.

   Therefore, Alice can verify that its original request was not altered before reception be the KDC, and because of the nonce, that this is not a replay of some previous request. In addition, the message includes two items intended for Bob:

   - The one-time session key K_s which is used for the session;
   - An identifier of Alice :ID_A.

   These last two items are encrypted with the master key that the KDC shares with Bob. They are to be sent to Bob to establish the connection and prove Alice's identity.

3. Alice stores the session key for use in the upcoming session and forwards to Bob the information that originated at the KDC for Bob, namely, $E_{K_B}[K_s||ID_A]$. Because this information is encrypted with K_B, it is protected from eavesdropping. Bob now knows the session key (K_s), knows that the other party is Alice (from ID_A), and that the information originated at the KDC(because it is encrypted using E_{K_B}).



Key Distribution Center (KDC)
: Key distribution steps
: Authentication steps
(1) Request || N_1
(2) E_{K_A} [K_S || Request || N_1 || E_{K_B}(K_S, ID_A)]
(3) E_{K_B}[K_S || ID_A]
Initiator
Alice
(4) E_{K_S}[N_2]
Responder
Bob
(5) E_{K_S}[f(N_2)]

Figure 3.3: Authenticated key distribution system.

At this point, a session key has been securely delivered to Alice and Bob, and they may begin their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, Bob sends a nonce, N_2, to Alice.

5. Also using K_s, Alice responds with $f(N_2)$, where it is a function that performs some transformation on N_2.

These steps assure Bob that the original message it received (step 3) has not a replay.

Note that actual key distribution involves only step 1 through 3 but that step 4 and 5, as well as 3, perform an authentication function.

3.4.2 Public-Key Certificates

Public-key certificates are a vehicle by which public keys may be stored, distributed or forwarded over unsecured media without danger of undetectable manipulation. The objective is to make one entity's public key available to others such that its authenticity(i.e., its status as the true public key of that entity) and validity are verifiable.

The "*Certification Authority*"(CA) is a trusted third party whose signature on the certificate vouches for the public key bound to the subject entity. The significance of this binding (e.g., what the key may be used for) must be provided by additional means, such as an attribute certificate or policy statement. Within the certificate, the string which identifies the subject entity must be a unique name within the system (*distinguished name*), which the CA typically associates with a real-world entity. The CA requires its own signature key pair, the authentic public key of which is made available to each party upon registering as an authorized system user. This CA public key allows any system user, though certificate acquisition and verification, to transitively acquire trust in the authenticity of the public key in any certificate signed by that CA.

Creation of public-key certificates

Before creating a public-key certificate for Alice, the certification authority should take appropriate measures (relative to the security level required), typically non-cryptographic in nature, to verify the claimed identity of Alice and the fact that public key to be certified is actually that of Alice. Two cases may be distinguished:

1. *trusted party creates key pair.* The trusted party creates a public-key pair, assign it to a specific entity, and includes the public key and the identity of that entity in the certificate. The entity obtains a copy of the corresponding private key over a secure (authenticate and private) channel after providing its identity. All parties subsequently using this certificate essentially delegate trust to this prior verification of identity by the trusted party.

2. *entity creates own key pair.* The entity creates its own public-key pair, and securely transfers the public key to the trusted party in a manner which preserves authenticity(e.g., over a trusted channel, or in person). Upon verification of the authenticity (source) of the public key, the trusted party creates the public-key certificate as above.

Use and Verification of public-key certificates

The overall process whereby Bob uses a public-key certificate to obtain the authentic public key of Alice may be summarized as follows:

1. (One time) acquire the authentic public-key of the certification authority.

2. Obtain an identifying string which uniquely identifies the intended party Alice.

3. Acquire over some unsecured channel (e.g. from a central public database of certificates, or from Alice directly), a public-key certificate corresponding to subject entity Alice and agreeing with the previous identifying string.

4. (a) Verify the current date and time against the validity period (id any) in the certificate, relying on a local trusted time/day-clock;
   (b) Verify the current validity of the CA's public key itself;
   (c) Verify the signature on Alice's certificate, using the CA's public key;
   (d) Verify that the certificate has not been revoked.

5. If all checks succeed, accept the public key in the certificate as Alice's authentic key.

3.5 Identity-Based Cryptosystem

3.5.1 Basic Idea and Analysis

An identity-based cryptographic system, which was proposed by A.Shamir [3], is an asymmetric system wherein an entity's public identification information (unique name) plays the role of its public key. This system is used as input by a trusted center T to compute the entity's corresponding private key.

In usual public-key cryptosystems, every user has a key-pair (s, P), where s is a secret key, only known to this user and P is a public key which anybody may know. By definition, public keys need not be protected for confidentiality; on the contrary, they have to be as made as public as possible. But this "publicity" becomes drawback toward active attacks, such as the substitution of a "false" public key to a "true" one in a directory. Therefore besides key-pair (s, P), we must include his identification string I and "guarantee" G that P is really the public key of user I, and not the one of an imposer I.

When we adopt the Identity-based systems, the public key is equivalent to the identity. i.e. $P = I$. And guarantee is equivalent to the secret key itself. i.e. $G = s$. Since there is no certificate to store and to check, this system has good properties.

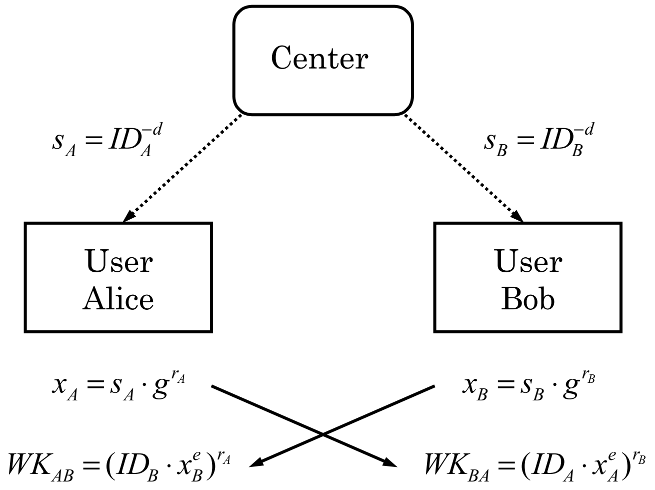
After computing the entity's private key, T transfers the entity's private key to the entity over a secure (authentic and private) cannel. This private key is computed from not only the entity's identity information, but must also be a function of some privileged information known only to T (T's private key). This is necessary to prevent forgery and impersonation - it is essential that only T be able to create valid private keys corresponding to give identification information. Corresponding (authentic) publicly available system data must be incorporated in the cryptographic transformations of the ID-based system, analogous to the certification authority public key in certificate-based systems.

3.5.2 Okamoto-Tanaka Key Exchange Scheme

This section summarizes the ID-KDS which was proposed by E.Okamoto and K.Tanaka [8]. This scheme consists of the following three phases.

Preparation phase

A trusted center picks two primes p and q, and makes n, g and e public, where $n = pq$, g is a generator of both Z_p^* and Z_q^*, and $e \in Z_{\lambda(n)}^*$. The Carmichael function of n is given by $\lambda(n) = \text{lcm}\,(p-1, q-1)$. Let $d \in Z_{\lambda(n)}^*$ be the secret key of the center satisfying $ed = 1$ (mod $\lambda(n)$).



Center
$s_A = ID_A^{-d}$
$s_B = ID_B^{-d}$
User Alice
User Bob
$x_A = s_A \cdot g^{r_A}$
$x_B = s_B \cdot g^{r_B}$
$WK_{AB} = (ID_B \cdot x_B^e)^{r_A}$
$WK_{BA} = (ID_A \cdot x_A^e)^{r_B}$

Figure 3.4: Okamoto-Tanaka key exchange scheme.

User's participation phase

Let ID_i be the user i's $(i = A, B, C, \ldots)$ identity information. Let s_i be the secret key of the user i satisfying

$$s_i = ID_i^{-d} \pmod{n}.$$

The center then publishes (e, n, g, ID_i) and delivers s_i to each user i through a secure channel or by using an IC card.

Common key generation phase

We assume here that two users Alice and Bob want to share a common-key. First, Alice generates a random number r_A, computes

$$x_A = s_A \cdot g^{r_A} \pmod{n},$$

and sends it to Bob. Similarly, Bob generates a random number r_B, computes

$$x_B = s_B \cdot g^{r_B} \pmod{n},$$

and sends it to Alice. Next, Alice computes

$$WK_{AB} = (ID_B \cdot x_B^e)^{r_A} \pmod{n}.$$

Similarly, Bob computes

$$WK_{BA} = (ID_A \cdot x_A^e)^{r_B} \pmod{n}.$$

WK_{AB} and WK_{BA} serve as a common key since

$$\begin{aligned} WK_{AB} &= (ID_B \cdot x_B^e)^{r_A} \\ &= (ID_B \cdot (s_B \cdot g^{r_B})^e)^{r_A} \\ &= (ID_B \cdot (ID_B^{-d})^e \cdot g^{r_B e})^{r_A} \\ &= g^{e r_A r_B} \\ &= WK_{BA} \pmod{n}. \end{aligned}$$

Chapter 4

Computational Complexity Theory

4.1 Complexity Issues in Cryptography

One very important observation is that a public-key cryptosystem can never provide unconditional security. Consequently, it is important to study the computational security of public-key systems.

Certainly a minimal expectation of a public-key cryptoystem is that this system cannot be cracked in polynomial time. If this condition is satisfied, then every polynomial time-bounded algorithm there exits infinitely many messages whose codes the algorithm, cannot crack. This leaves open possibility that there exists some algorithm, and infinitely many messages whose codes this algorithm can crack in polynomial time. For a public-key cryptoystem to be secure, it must be the case that ciphertexts cannot be cracked for most messages.

4.2 Turing Machine

We show a simple, yet powerful computing device called "*Turing machine*", which was invited by A.Turing. A Turing machine consists of two major components, a tape and a control unit. The "*Tape*" is a sequence of cells that extends to infinity in both directions. Each cell contains a symbol from a finite alphabet. There is a tape head that reads from a cell and writes into the same cell. The "*control unit*" contains a finite set of instructions, and it executes these instructions as follows: Each instruction causes the tape head to read the symbol from a cell, to write a symbol into the same cell, and either to move the tape head to an adjacent cell or to leave it at the same cell.

Each instruction of a Turing machine can be represented as a 5-tuple consisting of the following five parts:

1. The Turing machine state.

2. A tape symbol read from the current tape cell.

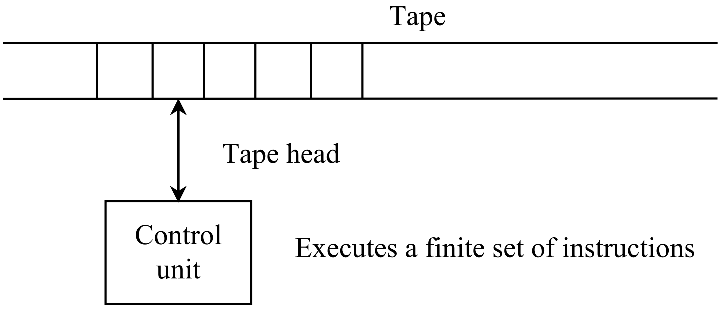


Figure 4.1: Overview of Turing machine.

3. A tape symbol to write into the current tape cell.
4. A direction for the tape head to move.
5. The next machine state.

We'll agree to let the letters L, S and R mean "*move left one cell,*", "*stay at the current cell*" and "*move right one cell,*" respectively. For example, suppose we have the following instruction:

$$< i, a, b, L, j > .$$

The instruction is interpreted as follows:

> If the current state of the machine is i, and if the symbol in the current tape cell is a, then write b into current tape cell, move left one cell, and goto state j.

The tape is used much like the memory in a modern computer, to store the input, to store data needed during execution, we need to make a few more assumptions:

1. An input string is represented on the tape by placing the letters of the string in contiguous tape cells. All other cells of the tape contain the blank symbol.
2. The tape head is positioned at the leftmost cell of the input string unless specified otherwise.
3. There is one "*start state*".
4. There is one "*halt state*".

The execution of a Turing machine stops when it enters the Halt state or when it enters a tate for which there is no valid move. For example, if a Turing machine enters state j and reads a in the current cell, but there is no instruction of the form $< i, a, \ldots >$, then the machine in state i.

We say that an input string is "*accepted*" by a Turing machine if the machine enters the Halt state. Otherwise, the input string is "*rejected*". There are two ways to reject an input string: Either the machine stops by entering a state other than the Halt state from which there is no move, or the machine runs forever. The *Turing machine* is the set of all input strings accepted by the machine.

It's easy to see that Turing machines can solve all the problem that "*Pushdown automatons*" which mean a finite automaton with a stack, can solve because a stack can be maintained on some portion of the tape. In fact, a Turing machine can maintain any number of stacks on the tape by allocating some space on the tape for each stack.

4.3 Complexity Classes

We first define the following definition.

Definition 4.3.1 [Polynomial-Time Algorithm] An algorithm whose worst-case running time function is of the form $O(n^k)$, where n is the input size and k is a constant, is called an *polynomial-time algorithm.*

Roughly speaking, polynomial-time algorithms can be equipped with "*good*" or "*efficient*" algorithms. When considering polynomial-time complexity, the degree of the polynomial is significant. For example, even though an algorithm with a running time of $O(n^{\ln \ln n})$, n being the input size, is asymptotically slower that an algorithm with a running time of $O(n^{100})$, the former algorithm may be faster in practice for smaller value of n, especially if the contains hidden by the big-O notion are smaller. Furthermore, in cryptography, average-case complexity is more important than worst-case complexity – a necessary condition for an encryption scheme to be considered secure is that the corresponding cryptanalysis problem is difficult on average (or better yet, always difficult), and not just for some isolated cases.

With respect to the complexity class, we show the following definition:

Definition 4.3.2 [Complexity Class P] The set of all decision problems that are solvable in polynomial time is called the *complexity class* **P**.

The computational tasks that need to get solved in practice are not all of the kind that take a "*yes*" or "*no*" answer. For example, we say need to find a satisfying truth assignment of a Boolean expression, not just to tell whether the expression is satisfiable; in the traveling salesman problem we want the optimal tour, not just whether a tour within a given budget exists; and so on. We call such problems requiring an answer more elaborate than "*yes*" or "*no*", "*function problems*". We can show the following definition which is associated with this problems.

Definition 4.3.3 [Complexity Class FP] The class of all function problems in **P** is called the *complexity class* **FP**.

4.4 Ordering among Difficulty of Functions

We show the methods of reductions among functions at first. These reductions are defined in the same way as the reductions among languages over some finite alphabet.

We define the function Q_1 to solve the program P_1, and the function q_2 to solve the program P_2, respectively. If one adopt P_1 as a subroutine, and can construct polynomial time computable program P_2, then we say *"Q_2 reduces to Q_1"*.

We show the instruction in graphical as follows:

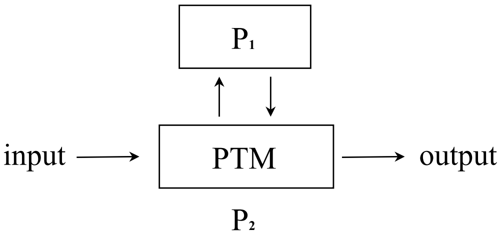


Figure 4.2: Querying an oracle.

In Figure 4.2, PTM means "*Polynomial-time Turing Machine*". With respect to reductions, there exist some classifications. At first, we define "*polynomial-time many-one reducibility*" as follows:

Definition 4.4.1 [Polynomial-Time Many-One Reducibility] For functions F and G, if there exists a pair of polynomial-time computable function (h_1, h_2) such that $F(x) = h_1(G(h_2(x)))$, then we say *"F reduces to G with respect to the polynomial-time many-one reducibility"*, and write as $F \leq_m^{FP} G$. If the converse reduction also holds, we say *"F is equivalent to G with respect to the polynomial-time many-one reducibility"*, and write as $F \equiv_m^{FP} G$.

In short, Definition 4.4.1 means that PTM is allowed to ask (only) one question to a oracle G, and on input the answer $(G(h_2(x))$ which the oracle G generate, outputs the value $h_1(G(h_2(x)))$ by using the function h_2. For simplicity, we show the figure as follows:

Proposition 4.4.2 Binary relation $F \leq_m^{FP} G$ is reflexive and transitive. That is:

(1) $F \leq_m^{FP} F$.

(2) If $F \leq_m^{FP} G$ and $G \leq_m^{FP} H$, then $F \leq_m^{FP} H$.

Proof.

(1) It is obvious.

(2) Because of $F \leq_m^{FP} G$ and $G \leq_m^{FP} H$, there exists two polynomial-time computable function f and g, and

$$x \in F \Leftrightarrow f(x) \in G \text{ ,and } x \in G \Leftrightarrow g(x) \in H$$

Therefore, $x \in A \Leftrightarrow g(f(x)) \in C$. This means $F \leq_m^{FP} H$.

■

Since Relation $F \equiv_m^{FP} G$ is reflexive, symmetric and transitive, this relation is equivalence.

In the case of many-one reducibility, this reduction restricts the question toward oracle for only one time.

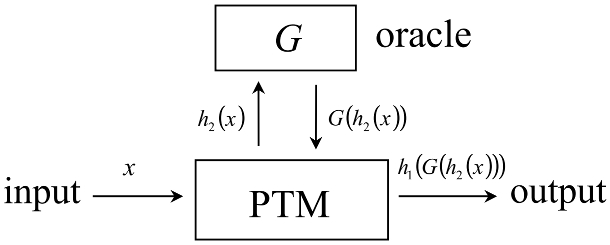


Figure 4.3: Many-one reducibility.

Next, we show the following two definition which was diminished this restriction.

Definition 4.4.3 [Polynomial-Time Turing Reducibility] For functions F and G, if one can construct a polynomial-time oracle Turing machine with access to value of g, which computes f, we say *"F reduces to G with respect to the polynomial-time Turing reducibility "*, and write as $F \leq_T^{FP} G$. Regarding the complexity of such an algorithm, we suppose that the cost of one calling the oracle is just one step. If the converse reduction also holds, we say *"F is equivalent to G with respect to the polynomial-time Turing reducibility"*, and write as $F \equiv_T^{FP} G$.

Definition 4.4.4 [Expected Polynomial-Time Turing Reducibility] For functions F and G, if an expected polynomial-time oracle Turing machine with access to values of g, can compute f, we say *"F reduces to G with respect to the expected polynomial-time Turing reducibility"*, and write as $F \leq_T^{EFP} G$. Here we say that a machine M is expected polynomial-time if there exists an $e > 0$ such that, for all $x \in \{0, 1\}$, the expectation, taken over the infinite bit sequences r, of $(t_M(x, r))^e$ is bounded above by $|x|$, *i.e.*,

$$E((t_M(x, r))^e) \leq |x|.$$

As seen in the definitions, if $F \leq_m^{FP} G$ or $F \leq_T^{FP} G$ holds, and if the polynomial-time algorithm for computing G is discovered, then so is that for computing F. For two functions F and G, $F \leq_m^{FP} G$ implies $F \leq_T^{FP} G$, and that $F \leq_T^{FP} G$ implies $F \leq_T^{EFP} G$.

4.5 Functions to Break Protocols

We first define functions related with the several problems.

Definition 4.5.1 **[Discrete Logarithm Problem]** `DLP`(n, g, y) is a function that on input $n \in N_{>1}$, $g \in Z_n^*$, $y \in Z_n^*$, outputs w such that $y = g^w \pmod{n}$ and $0 \leq w < n$, if such an w exists.

Definition 4.5.2 **[Factoring Problem]** `Factoring`(n) is a function that on input $n \in N_{>1}$, outputs a such that $1 < a < n$ and $a|n$, if such an a exists.

Now it is not known whether complexity class **FP** contains the discrete logarithm problem or not. the same statement also holds for the factoring problem.

Note that the oracle has no responsibilities when the input is inappropriate. However, we will assume that on correct inputs the oracle will return an answer within some polynomial time bound. For example, suppose B is an oracle for the function `DLP`. When $n = pq$ is not the product of two prime, or g is not a generator of Z_n^*:

1. A may return a syntactically incorrect answer; in which case we know that either p or g is a inappropriate.
2. A may fail to give an answer within a given polynomial time bound, and again we conclude that either n or g is inappropriate. (Here we assume that the time bound can be computed in polynomial time.)
3. A may return a syntactically correct answer. The answer can be checked to see if it satisfies the appropriate equivalence, but we cannot conclude that n is the product of two primes p and q or g is a generator.

Next, we define functions to break the several protocols which are all based on discrete logarithm problem. We assume that the base g is in Z_n^*. This assumption includes the case of g being a primitive root modulo both p and q.

Definition 4.5.3 **[Diffie-Hellman Key Exchange Scheme]** `DH`(n, g, y_A, y_B) is a function that on input $n \in N_{>1}$, $g \in Z_n^*$, $y_A \in Z_n^*$, $y_B \in Z_n^*$, outputs $K \in Z_n^*$ such that $K = g^{ab} \pmod{n}$, where $y_A = g^a \pmod{n}$ and $y_B = g^b \pmod{n}$, if such a K exists.

Definition 4.5.4 **[Okamoto-Tanaka Key Exchange Scheme]** `OT`$(n, e, g, ID_A, ID_B,$ x_A, $x_B)$ is a function that on input $n \in N_{>1}$, $e \in Z_{\lambda(n)}^*$, $g \in Z_n^*, ID_A \in Z_n^*$, $ID_B \in Z_n^*$, $x_A \in Z_n^*, x_B \in Z_n^*$, outputs $WK \in Z_n^*$ such that $WK = (ID_B \cdot x_B^e)^{r_A} = (ID_A \cdot x_A^e)^{r_B} = g^{er_Ar_B} \pmod{n}$, where $x_A = g^{r_A} \cdot (ID_A)^{-e^{-1}} \pmod{n}$, $x_B = g^{r_B} \cdot (ID_B)^{-e^{-1}} \pmod{n}$ and $ee^{-1} = e^{-1}e = 1 \pmod{\lambda(n)}$, if such a WK exists.

Other functions based on several problems are defined as follows:

Definition 4.5.5 [RSA Public-Key Cryptosystem] $\mathtt{RSA}(n,e,y)$ is a function that on input $n \in N_{>1}$, $e \in Z^*_{\lambda(n)}$, $y \in Z^*_n$, outputs $x \in Z^*_n$ such that $y = x^e \pmod{n}$, if such an x exists.

Definition 4.5.6 [ElGamal Public-Key Cryptosystem] $\mathtt{EG}(n,g,y,C_1,C_2)$ is a function that on input $n \in N_{>1}$, $g \in Z^*_n$, $y \in Z^*_n$, $C_1 \in Z^*_n$, $C_2 \in Z^*_n$, outputs $m \in Z^*_n$ such that $m = C_2/C_1^x \pmod{n}$, where $y = g^x \pmod{n}$, if such an m exists.

4.6 Reducibility among Functions

4.6.1 Difficulty of Breaking Key Sharing

With respect to the key sharing of the Okamoto-Tanaka scheme, we indicate the following theorem.

Theorem 4.6.1 $\mathtt{OT} \equiv^{FP}_m \mathtt{DH}$.

Proof.

1. $\mathtt{OT} \leq^{FP}_m \mathtt{DH}$:

$$\mathtt{OT}(n,e,g,ID_A,ID_B,x_A,x_B) = \mathtt{DH}(n,e,g^e,ID_Ax_A^e,ID_Bx_B^e).$$

2. $\mathtt{DH} \leq^{FP}_m \mathtt{OT}$:

$$\mathtt{DH}(n,g,y_A,y_B) = \mathtt{OT}(n,1,g,1,1,y_A,y_B).$$

■

In Theorem 4.6.1, note that as follows:

1. This reduction holds whenever g is in Z^*_n. Therefore, there is no need for us to assume that g is a primitive root modulo both p and q. Note that the order of a subgroup generated by a common primitive root is $\mathrm{lcm}(p-1,q-1) \leq \varphi(n)/2$, where φ is the Euler totient function.

2. In the proof of $\mathtt{DH} \leq^{FP}_m \mathtt{OT}$, we put $e = 1 \in Z^*_{\varphi(n)}$ and $ID_A = ID_B = 1 \in Z^*_n$. If ID_A and ID_B must be different from each other, one may pick any two distinct ID_A and ID_B from Z^*_n, and generate a query to be $(n,1,g,ID_A,ID_B,y_A/ID_A,y_B/ID_B)$ so that $\mathtt{OT}$ returns $WK = g^{ab} \pmod{n}$. Moreover, if for e such that $e = 1$ or $e \notin Z^*_n$, $\mathtt{OT}$ returns $\perp$, then the reduction algorithm becomes something else. The following is a probabilistic polynomial-time algorithm that reduces $\mathtt{DH}$ to $\mathtt{OT}$.

One may pick any odd e from $Z^*_n \setminus \{1\}$, choose s_A and s_B from Z^*_n, and generate a query to be $(n,e,g,s_A^e y_A, s_B^e, s_A^{-1}g, s_B^{-1}y_B)$. If e is in fact in $Z^*_{\varphi(n)} \setminus \{1\}$, then $\mathtt{OT}$

returns a correct $WK = g^{ab+eb}$. This is because $s_A^e A(s_A^{-1} g)^e = y_A g^e = y_A g^e = g^{a+e} = g^{ee^{-1}(a+e)}$, $s_B^e(s_B^{-1} y_B) = y_B^e = g^{eb}$, and so $WK = g^{ee^{-1}(a+e)b} = g^{(ab+eb)}$. Therefore,

$$\mathtt{DH}(n, g, y_A, y_B) = \mathtt{OT}(n, e, g, s_A^e y_A, s_B^e, s_A^{-1} g, s_A^{-1} y_B)/y_B^e \pmod{n},$$

if e is in $Z^*_{\varphi(n)} \setminus \{1\}$. On the other hand, if $e = 1$ or not in $Z^*_{\varphi(n)}$, OT returns $\perp$.

The probability that e picked in such a way is in $Z^*_{\varphi(n)}$ is estimated as

$$\rho = \frac{\varphi(\varphi(n))}{\varphi(n) - \dfrac{n-1}{2} - 1} \geq \frac{2\varphi(\varphi(n))}{\varphi(n)} \geq \frac{2\ln(2)\varphi(n)}{\varphi(n)\ln(2\varphi(n))} \geq \frac{2\ln(2)}{\ln(2n)},$$

where we used an inequality that $\varphi(n) \geq \ln(2) \cdot n/\ln(2)$ for a positive integer n. Thus if e is picked repeatedly for $\ln(2n)/(2)\ln(2)$times, one may expect that there is at least one correct answer other than $\perp$ among the values returned from OT. The expected number of repetition is bounded above by a polynomial in $|n|$. Therefore the reduction algorithm runs in expected polynomial time.

Specifically, if the modulus n is chosen such that $n = pq$, $p = 2p' + 1$, $q = 2q' + 1$ with p, q, p', q' prime, then

$$\rho \geq \frac{2\varphi(\varphi(n))}{\varphi(n)} = 1 - \frac{1}{p'} - \frac{1}{p'q'}.$$

This means that the reduction algorithm works in a deterministic matter with overwhelming probability.

Next we show that EG is equivalent to DH as follows:

Theorem 4.6.2 EG $\equiv^p_m$ DH.

Proof.

1. DH $\leq^{FP}_m$ EG:

$$\mathtt{DH}(n, g, y_A, y_B) = \mathtt{EG}(n, g, y_A^{-1}, B, 1),$$

   where y_A^{-1} is the inverse of $y_A \pmod{n}$. Note that

$$\mathtt{EG}(n, g, y_A^{-1}, y_B, 1) = \mathtt{EG}(n, g, g^{-a}, g^b, g^{ab-ba}).$$

2. EG $\leq^{FP}_m$ DH:

$$\mathtt{EG}(n, g, y, C_1, C_2) = C_2/\mathtt{DH}(n, g, y, C_1) \pmod{n}.$$

■

Besides this reductions, it is known at present that DH $\leq^{FP}_m$ DLP, RSA $\leq^{FP}_m$ Factoring, and Factoring $\leq^{EFP}_m$ DLP in [22] and [23].

4.6.2 Difficulty of Impersonation

We show the security with respect to the impersonation of the Okamoto-Tanaka scheme. This scheme use the RSA scheme for identification checking. Therefore, it is natural for us to expect that the security would depend on the RSA. However, Theorem 4.6.1 shows that breaking the function $\mathtt{OT}$ is equivalent to breaking the function $\mathtt{DH}$. This means nothing about relationship between $\mathtt{OT}$ and $\mathtt{RSA}$.

In fact if $\mathtt{RSA}$ were a polynomial-time computable function, anyone could break identification, that is, one could compute $s_A = ID_A^{-e^{-1}} \pmod n$ by $\mathtt{RSA}(n, e, 1/ID_A)$, and impersonate Alice or Bob. Therefore, computing s_A or s_B from (n, e, ID_A, ID_B) reduces to computing $\mathtt{RSA}$ with respect to polynomial-time many-one reducibility. This indicate the security against impersonation depends on the difficulty of computing $\mathtt{RSA}$. However, we indicate that even $\mathtt{RSA}$ reduces to $\mathtt{DH}$. This implies that impersonation is easier than breaking.

Theorem 4.6.3 $\mathtt{RSA} \leq_m^{FP} \mathtt{DH}$.

Proof.

$$\mathtt{RSA}(n, e, y) = \mathtt{DH}(n, y^e, y, y).$$

Note that

$$\mathtt{DH}(n, y^e, y, y) \quad = \quad \mathtt{DH}(n, y^e, y^{ed}, y^{ed}) \quad = y^{ed^2} \quad = y^d \quad = x \pmod n,$$

where $ed = de = 1 \pmod{\varphi(n)}$.

∎

4.7 Concluding Remarks on Reductions

As a consequence, we can summarize the reductions related with $\mathtt{DLP}$ as follows:

1. $\mathtt{RSA} \leq_m^{FP} \mathtt{OT} \equiv_m^{FP} \mathtt{DH} \equiv_m^{FP} \mathtt{EG} \leq_m^{FP} \mathtt{DLP}$.
2. $\mathtt{RSA} \leq_m^{FP} \mathtt{Factoring} \leq_T^{FP} \mathtt{DLP}$.

It is not known at present to hold that $\mathtt{DH}$ reduces to $\mathtt{RSA}$, $\mathtt{DLP}$ reduces to $\mathtt{DH}$, or $\mathtt{Factoring}$ reduces to $\mathtt{DH}$, with respect to some polynomial-time reducibility. These remain as open questions.

Chapter 5

Fault-Tolerant Key Distribution Systems

5.1 Concept and Analysis

5.1.1 Basic Idea

In ID-KDS, many useful schemes [8] - [12] are proposed up to now. These schemes are efficient for implementation, but they have certain drawbacks at the stage in which the center revoke and renew a users secret information. That is, when user's secret information is made public for some reason, the user must ask for the center to revoke it and get a new one. When we implement ID-KDS on networks, it is preferable that the center adopts one uniform ID such as a user's name, an e-mail address, a social security number, and so on, and never change that ID for any reason. But unfortunately, these schemes cannot satisfy above condition. To solve these problems, we show the following definition.

Definition 5.1.1 [Fault-Tolerant Key Distribution System] A system which satisfy the following conditions 1-3 is called *Fault-Tolerant Key Distribution System.*

1. This system is a centralized key management system which exists a certification authority (CA).
2. This system can generate plural signatures for a message and each signature is different from another one.
3. If two or more users which take part in this system want to share the common key, each user can use the signature as authentication.

By using this system, even after a center revoke the user's secret information, the center can generate a new one without any change of ID. This means there is no need for the user to make a file which contains several pieces of ID for one user. Therefore, this system can diminish the burden of the user in this sense.

5.1.2 Levels of Trust

In 1991, M.Giraut [20] classified the levels of trust for centralized key management. With respect to the fault-tolerant key distribution system, we can reconstruct this level as follows:

1. The authority knows (or can easily compute) user's all secret information. Therefore, the authority can impersonate any user at any time without detected.
2. The authority does not know(or cannot easily compute) user's secret information. Nevertheless, the authority can still impersonate a user by generating false guarantees.
3. Even after the authority generates the false guarantees, the impersonation of the authority can be detective.

Ideally, the level 3 is the most desirable one. However in this case, we need another authority which guarantee a user's secret information. Therefore, when it comes to implement the centralized key management system, we must consider this problem.

5.2 Proposed Scheme

This section shows our proposed scheme. There are three kinds of protocols, and each protocol has one or several phase(s) in this scheme.

5.2.1 Key Sharing Protocol

In this section, we discuss the key sharing protocol by which two users can agree on a common key. This protocol consists of the following three phases.

Preparation phase

A trusted center picks two primes p and q, and makes n, g and e public, where $n = pq$, g is a generator of both Z_p^* and Z_q^*, and $e \in Z_{\lambda(n)}^*$. The Carmichael function of n is given by $\lambda(n) = \mathrm{lcm}\,(p-1, q-1)$. Let $d \in Z_{\lambda(n)}^*$ be the secret key of the center satisfying $ed = 1 \pmod{\lambda(n)}$.

User's participation phase

Let ID_i be the user i's $(i = A, B, C, \ldots)$ identity information. Let f be a one-way hash function which maps from ID_i to the following m-dimensional vector(ID-vector) $\mathbf{v}_i$, as

$$\mathbf{v}_i = f(ID_i) = (v_{i0}, v_{i1}, v_{i2}, \ldots, v_{i(m-1)}),$$

where $v_{ij} \neq v_{ij'}$ if $j \neq j'$ for all j, and $v_{ij} \in Z_n$, $(0 \leq j \leq m-1)$.

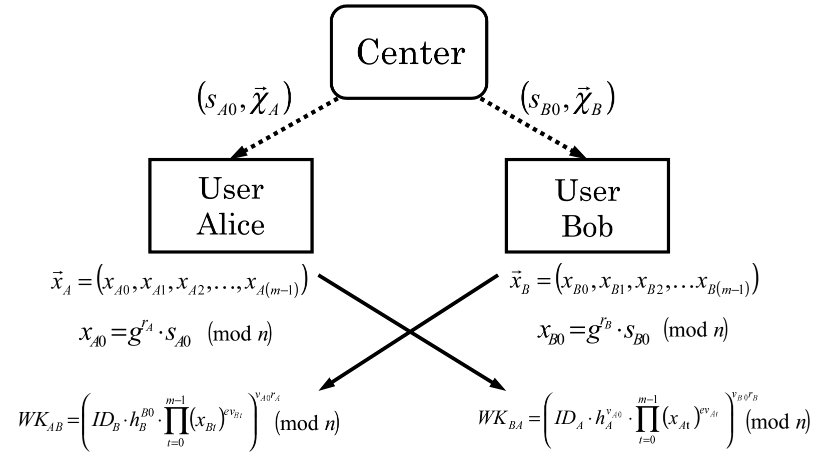

Center
$(s_{A0}, \vec{\chi}_A)$
$(s_{B0}, \vec{\chi}_B)$
User Alice
User Bob
$\vec{x}_A = (x_{A0}, x_{A1}, x_{A2}, \ldots, x_{A(m-1)})$
$x_{A0} = g^{r_A} \cdot s_{A0} \pmod n$
$\vec{x}_B = (x_{B0}, x_{B1}, x_{B2}, \ldots x_{B(m-1)})$
$x_{B0} = g^{r_B} \cdot s_{B0} \pmod n$
$WK_{AB} = \left(ID_B \cdot h_B^{B0} \cdot \prod_{t=0}^{m-1} (x_{Bt})^{ev_{Bt}} \right)^{v_{A0} r_A} \pmod n$
$WK_{BA} = \left(ID_A \cdot h_A^{v_{A0}} \cdot \prod_{t=0}^{m-1} (x_{At})^{ev_{At}} \right)^{v_{B0} r_B} \pmod n$

Figure 5.1: Proposed scheme.

The center generates m random numbers k_{i0}, k_{i1}, k_{i2}, ... , $k_{i(m-1)}$ (Let $\mathbf{k}_i$ be (k_{i0}, k_{i1}, k_{i2}, ... , $k_{i(m-1)}$)), and then computes σ_i, τ_i, and χ_i as follows:

$$\tau_i = \mathbf{k}_i \cdot \mathbf{v}_i = \sum_{t=0}^{m-1} k_{it} v_{it} \pmod{\lambda(n)},$$

$$\sigma_i \tau_i = 1 \pmod{\lambda(n)}, \quad \text{and}$$

$$\chi_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{i(m-1)}),$$

where

$$x_{ij} = ID_i^{-d\sigma_i k_{ij}} (\text{mod } n),$$

for each j, $(1 \leq j \leq m-1)$.

Let h be a one-way hash function which computes h_i and the center computes s_{i0}, i.e.

$$h_i = h(x_{i1}, x_{i2}, x_{i3}, \ldots, x_{i(m-1)}), \quad \text{and}$$

$$s_{i0} = (h_i \cdot ID_i^{\sigma_i k_{i0}})^{-d}.$$

The center then publishes (e, n, g, ID_i, f, h) and delivers (s_{i0}, χ_i) to each user i. With respect to s_{i0}, it must be delivered through a secure channel or by using an IC card.

Common key generation phase

We assume here that two users Alice and Bob want to share a common-key. First, Alice generates a random number r_A and computes

$$x_{A0} = g^{r_A} \cdot s_{A0} \pmod n.$$

We define

$$\mathbf{x}_A = \bigl(x_{A0}, x_{A1}, x_{A2}, \ldots, x_{A(m-1)}\bigr),$$

and Alice sends it to Bob. Similarly, Bob generates a random number r_B and computes

$$x_{B0} = g^{r_B} \cdot s_{B0} \pmod{n}.$$

We define

$$\mathbf{x}_B = \bigl(x_{B0}, x_{B1}, x_{B2}, \ldots, x_{B(m-1)}\bigr),$$

and Bob sends it to Alice. Then Alice computes

$$WK_{AB} = \biggl(ID_B \cdot h_B^{v_{B0}} \cdot \prod_{t=0}^{m-1} (x_{Bt})^{ev_{Bt}}\biggr)^{v_{A0} r_A} \pmod{n}.$$

Similarly, Bob computes

$$WK_{BA} = \biggl(ID_A \cdot h_A^{v_{A0}} \cdot \prod_{t=0}^{m-1} (x_{At})^{ev_{At}}\biggr)^{v_{B0} r_B} \pmod{n}.$$

WK_{AB} and WK_{BA} serve as a common key since

$$\begin{aligned}
WK_{AB} &= \biggl(ID_B \cdot h_B^{v_{B0}} \cdot \prod_{t=0}^{m-1} (x_{Bt})^{ev_{Bt}}\biggr)^{v_{A0} r_A} \\
&= \biggl(ID_B \cdot h_B^{v_{B0}} \cdot (g^{v_{B0} e r_B} \cdot h_B^{-edv_{B0}}) \\
&\qquad \cdot \prod_{t=0}^{m-1} (ID_B^{-d\sigma_B k_{Bt}})^{ev_{Bt}}\biggr)^{v_{A0} r_A} \\
&= \biggl(ID_B ID_B{}^{-ed\sigma_B\left(\sum_{t=0}^{m-1} v_{At} k_{At}\right)} \\
&\qquad \cdot h_A^{v_{B0}} \cdot h_A^{-edv_{B0}} \cdot g^{v_{B0} e r_B}\biggr)^{v_{A0} r_A} \\
&= g^{c_{AB} e r_A r_B} \\
&= WK_{BA} \pmod{n},
\end{aligned}$$

where $c_{AB} := v_{A0} \cdot v_{B0}$.

5.2.2 Revocation and Renewal Protocol

In this section, we discuss the revocation and renewal protocol for a specific user's information. This protocol consists of the following two phases.

Revocation phase for user's secret information

When the center revokes the secret information for a specific user i, the center sends the following information related with the user i to all users:

- i : the specific user which has been revoked, and
- h_i : the value of a hash function h which is computed from χ_i.

Renewal phase for user's secret information

The center newly generates m random numbers k'_{i0}, k'_{i1}, k'_{i2}, $\ldots$, $k'_{i(m-1)}$ (Let $\mathbf{k}'_i$ be $(k'_{i0}$, k'_{i1}, k'_{i2}, $\ldots$, $k'_{i(m-1)}))$, $k_{ij} \neq k'_{ij}$, for each j, $(0 \leq j \leq m-1)$, and then computes σ'_i, τ'_i, χ'_i, h'_i, and s'_{i0} as follows:

$$\tau'_i = \mathbf{k}'_i \cdot \mathbf{v}_i = \sum_{t=0}^{m-1} k'_{it} v_{it} \pmod{\lambda(n)},$$

$$\sigma'_i \tau'_i = 1 \pmod{\lambda(n)},$$

$$\chi'_i = (x'_{i1}, x'_{i2}, x'_{i3}, \ldots, x'_{i(m-1)}),$$

where

$$x'_{ij} = ID_i^{-d\sigma'_i k'_{ij}} (\bmod\ n),$$

for each j, $(1 \leq j \leq m-1)$,

$$h'_i = h(x'_{i1}, x'_{i2}, x'_{i3}, \ldots, x'_{i(m-1)}), \qquad \text{and}$$

$$s'_{i0} = (h'_i ID_i^{\sigma'_i k'_{i0}})^{-d}.$$

Note that the center computes τ'_i by using ID-vector $\mathbf{v}_i$. This means that even after the center has revoked a secret information s_{i0}, the center generates a new one s'_{i0} without any change of ID. This is the concept of our proposed scheme.

The center then delivers (s'_{i0}, χ'_i) to the specific user i. With respect to s'_{i0}, it must be delivered through a secure channel or by using an IC card.

5.2.3 New Key Sharing Protocol

In this section, we discuss the new key sharing protocol. If the center has executed the revocation and the renewal protocol in Sect.5.2.2, the user must include the following phases' protocol in the common key generation phase given in Sect.5.2.1. This protocol consists of the following phase.

New common key generation phase

We assume the center has already revoked Alice's information s_{A0} and χ_A the center delivers s'_{A0} and χ'_A to the user Alice. Hence Alice send $\mathbf{x}_A$ to Bob by using s'_{A0} and χ'_A.

When Bob gets $\mathbf{x}_i$ from Alice, Bob confirms whether the following verification holds:

$$h_A \overset{?}{\neq} h'_A \pmod{n}.$$

If it is true, Bob goes to the next step. Otherwise, Bob stops the protocol.

5.3 Security Considerations

As shown in Sect.3.5.2 and 5.2, the original scheme and the proposed one are of the same form. Hence these schemes are basically of the same type as the Diffie-Hellman scheme over Z_n. An additional operation for verifying the identity is based on RSA. Therefore, it is natural to conjecture that the security should be related to the security of Diffie-Hellman and that of RSA. In fact, it is obvious that if both Diffie-Hellman and RSA were easy to break, then so would be the original and the proposed schemes. Here, we consider the security by using polynomial-time reductions.

5.3.1 Functions to Break Protocols

We first define the function PROPOSAL to break the proposed key sharing schemes. We assume that the base g is just in Z_n^*. This assumption includes the case of g being a primitive root modulo both p and q, defined in the original and the proposed scheme.

Definition 5.3.1 $\mathtt{PROPOSAL}(n, e, g, h_A, h_B, ID_A, ID_B, \mathbf{v}_A, \mathbf{v}_B, \mathbf{x}_A, \mathbf{x}_B)$ is a function that on input $n \in N_{>1}$, $e \in Z_{\lambda(n)}^*$, $g \in Z_n^*$, $h_A \in Z_n^*$, $h_B \in Z_n^*$, $ID_A \in Z_n^*$, $ID_B \in Z_n^*$, $v_{Aj} \in Z_n, v_{Bj} \in Z_n$, $x_{Aj} \in Z_n^*$, $x_{Bj} \in Z_n^*$, outputs $WK \in Z_n^*$ such that

$$\begin{aligned} WK &= \left(ID_B \cdot h_B^{v_{B0}} \cdot \textstyle\prod_{t=0}^{m-1} (x_{Bt})^{ev_{Bt}}\right)^{v_{A0} r_A} \\ &= \left(ID_A \cdot h_A^{v_{A0}} \cdot \textstyle\prod_{t=0}^{m-1} (x_{At})^{ev_{At}}\right)^{v_{B0} r_B} \\ &= g^{c_{AB} e r_A r_B} \pmod{n}, \end{aligned}$$

where

$$\begin{aligned} c_{AB} &= v_{A0} \cdot v_{B0}, \\ h_A &= h(x_{A1}, x_{A2}, x_{A3}, \ldots, x_{A(m-1)}), \\ h_B &= h(x_{B1}, x_{B2}, x_{B3}, \ldots, x_{B(m-1)}), \\ \textstyle\prod_{t=0}^{m-1} (x_{At})^{v_{At}} &= (h_A^{v_{A0}} \cdot ID_A)^{-e^{-1}} \pmod{n}, \\ \textstyle\prod_{t=0}^{m-1} (x_{Bt})^{v_{Bt}} &= (h_B^{v_{B0}} \cdot ID_B)^{-e^{-1}} \pmod{n}, \qquad \text{and} \\ ee^{-1} &= e^{-1}e = 1 \pmod{\lambda(n)}, \end{aligned}$$

if such a WK exists.

5.3.2 Difficulty of Breaking Key Sharing

We prove the following theorem.

Theorem 5.3.2 $\mathtt{DH} \equiv_m^{FP} \mathtt{PROPOSAL}$.

Proof.

1. $\mathtt{DH} \leq_m^{FP} \mathtt{PROPOSAL}$

$$\mathtt{DH}(n, g, A, B) = \mathtt{PROPOSAL}(n, 1, g, 1, 1, 1, 1, \mathbf{1}_A^r, \mathbf{1}_B^r, \mathbf{1}_A^a, \mathbf{1}_B^b),$$

where $\mathbf{1}_A^r$, $\mathbf{1}_B^r$, $\mathbf{1}_A^a$, and $\mathbf{1}_B^b$ are the m-dimensional vectors defined as

$$\begin{aligned} \mathbf{1}_A^r &= (1, r_{A1}, r_{A2}\ldots, r_{A(m-1)}), \\ \mathbf{1}_B^r &= (1, r_{B1}, r_{B2}\ldots, r_{B(m-1)}), \\ \mathbf{1}_A^a &= (g^a, 1, 1\ldots, 1), \quad \text{and} \\ \mathbf{1}_B^b &= (g^b, 1, 1\ldots, 1), \end{aligned}$$

with $r_{Aj}, r_{Bj} \in_R Z_n$, for each j, $(1 \leq j \leq m-1)$.

2. $\texttt{PROPOSAL} \leq_m^{FP} \texttt{DH}$

$\prod \mathbf{x}_A^{e\mathbf{v}_A}$ and $\prod \mathbf{x}_B^{e\mathbf{v}_B}$ are defined as

$$\begin{aligned} \prod \mathbf{x}_A^{e\mathbf{v}_A} &= \textstyle\prod_{t=0}^{m-1} (x_{At})^{ev_{At}} \\ &= \left(g^{er_A} \cdot h^{-1}\right)^{v_{A0}} \cdot ID_A^{-\sigma_A\left(\sum_{t=0}^{m-1} k_{At}v_{At}\right)} \\ &= g^{v_{A0}er_A} \cdot h^{-v_{A0}} \cdot ID_A^{-1}, \quad \text{and} \\ \prod \mathbf{x}_B^{e\mathbf{v}_B} &= \textstyle\prod_{t=0}^{m-1} (x_{Bt})^{ev_{Bt}} \\ &= g^{v_{B0}er_B} \cdot h^{-v_{B0}} \cdot ID_B^{-1}. \end{aligned}$$

Note that m is a constant number. Thus both $\prod \mathbf{x}_A^{e\mathbf{v}_A}$ and $\prod \mathbf{x}_B^{e\mathbf{v}_B}$ are feasibly computable.

Therefore,

$$\begin{aligned} &\texttt{PROPOSAL}(n, e, g, h_A, h_B, ID_A, ID_B, \mathbf{v}_A, \mathbf{v}_B, \mathbf{x}_A, \mathbf{x}_B) \\ &\quad = \texttt{DH}(n, g^e, ID_A \cdot h_A^{v_{A0}} \cdot \textstyle\prod \mathbf{x}_A^{e\mathbf{v}_A}, ID_B \cdot h_B^{v_{B0}} \cdot \prod \mathbf{x}_B^{e\mathbf{v}_B}) \\ &\quad (= g^{c_{AB}er_Ar_B}). \end{aligned}$$

■

5.3.3 Difficulty of Impersonation

In our proposed scheme, RSA is used only for identification checking. In fact, if RSA were a polynomial-time computable function, anyone could break the identification. Therefore, computing s_{A0} or s_{B0} from the public information as $(n\ , e\ , g\ , h_A\ , h_B\ , ID_A\ , ID_B\ , \mathbf{v}_A\ , \mathbf{v}_B\ , \mathbf{x}_A\ , \mathbf{x}_B)$ reduces to computing RSA with respect to the polynomial-time many-one reducibility. In other words, the security against impersonation depends on the difficulty of computing RSA. However, it is not known whether the converse reduction also holds at present. The same statement holds for the original scheme.

In order to certify the security against impersonation, we must show that a shared key cannot be forged, even if collusion among users would be allowed, under the assumption that it is very difficult to factor n.

Here, we will show two considerable attacks and examine the security for them. We assume that the attacker Carol tries to impersonate Alice, and computes a common key between Carol and Bob.

Attack 1

Carol generates a random number $r_{A'}$ and m random numbers $k_{A'0}, k_{A'1}, k_{A'2}, \ldots, k_{A'(m-1)}$ (Let $\mathbf{k}_{A'}$ be $(k_{A'0}\ , k_{A'1}\ , k_{A'2}\ , \ldots\ , k_{A'(m-1)})$), and computes

$$\mathbf{x}_{A'} = (x_{A'0}, x_{A'1}, x_{A'2}, \ldots, x_{A'(m-1)}),$$

where

$$x_{A'j} = g^{k_{A'j}} \pmod{n},$$

for each j, $(0 \leq j \leq m-1)$, and sends it to Bob. Next, Carol gets $\mathbf{x}_B$ from Bob.

Carol and Bob compute

$$WK_{A'B} = \left(ID_B \cdot h_B^{v_{B0}} \cdot \prod_{t=0}^{m-1} (x_{Bt})^{ev_{Bt}}\right)^{v_{A0} r_{A'}} \pmod{n},$$

$$WK_{BA'} = \left(ID_A \cdot h_{A'}^{v_{A0}} \cdot \prod_{t=0}^{m-1} (x_{A't})^{ev_{At}}\right)^{v_{B0} r_B} \pmod{n},$$

respectively. But Carol can not get a common key since $WK_{A'B} \neq WK_{BA'}$.

Attack 2

We suppose the center has revoked Alice's secret information s_{A0}, and Carol knows (s_{A0}, χ_A). Then Carol computes

$$(h_A^{v_{A0}} \cdot ID_A)^{-d} = s_{A0}^{v_{A0}} \cdot \prod_{t=1}^{m-1} (x_{At})^{v_{At}} \pmod{n},$$

by using (s_{A0}, χ_A). Carol generates a random number $r_{A'}$ and computes

$$x_{A'0} = g^{r_{A'}} \cdot (h_A^{v_{A0}} \cdot ID_A)^{-d} \pmod{n}.$$

We define

$$\mathbf{x}_{A'} = (x_{A'0}, x_{A1}, x_{A2}, \ldots, x_{A(m-1)}),$$

and sends it to Bob. Next, Carol gets $\mathbf{x}_B$ from Bob.

Carol and Bob compute

$$WK_{A'B} = \left(ID_B \cdot h_B^{v_{B0}} \cdot \prod_{t=0}^{m-1} (x_{Bt})^{ev_{Bt}}\right)^{v_{A0} r_{A'}} \pmod{n},$$

$$WK_{BA'} = \left(ID_A \cdot h_{A'}^{v_{A0}} \cdot \prod_{t=0}^{m-1} (x_{A't})^{ev_{At}}\right)^{v_{B0} r_B} \pmod{n},$$

respectively. But Carol can not get a common key since $WK_{A'B} \neq WK_{BA'}$.

We can not find any other effective attacks against impersonate with respect to the proposed scheme.

5.4 Conceptual Analysis of the Proposed Scheme

In the original scheme, the center computes the user's secret information including the digital signature, which is based on a ID information. Furthermore, in the proposed scheme the center computes the value τ_i by using both the ID vector $\mathbf{v}_i$ and the m-dimensional vector $\mathbf{k}_i$ which consists of m random numbers. From τ_i, the center computes σ_i which is the inverse of τ_i on the exponent, and the m-dimensional vector $((ID_i^{\sigma_i k_{A0}})^{-d}$, $\chi_i)$. Note that each value of this vector is signed digitally.

Then the center computes h_i from χ_i, signs this value digitally such as h_i^{-d}, and delivers $s_{i0}(= h_i^{-d} \cdot (ID_i^{\sigma_i k_{A0}})^{-d})$ as a secret information of each user. Therefore, each user does not know the values of h_i^{-d} and ID_i^{-d}.

The user's secret information works as authentication of each user. Therefore, by generating new random numbers, the center can newly generate several pieces of secret information for each user. This property enables the center to revoke and renew the user's secret information.

Both the original and the proposed schemes have several advantages as follows:

1: There is no need for the user to change n for the modulo, although the RSA cryptosystem does not have this property.

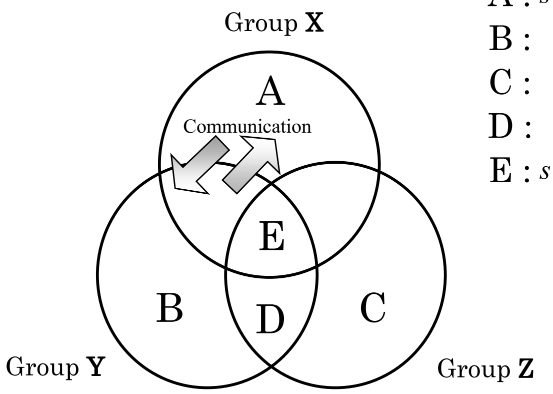
2: The user can obtain a different work key in each distribution.

Moreover, the proposed scheme has the following additional advantages:

3: The center can revoke and renew a user's secret information without any change of ID.

4: After the center has revoked a specific user's information, the other users do not need to change their secret information.

5: The revoked information can be distributed to all users within a constant time which does not depend on total numbers of users.



	X	Y	Z
A :	s_{XA}		
B :		s_{YB}	
C :			s_{ZC}
D :		s_{YD}	s_{ZD}
E :	s_{XE}	s_{YE}	s_{ZE}

Figure 5.2: Example of group communication system.

5.5 Applications to Other Key Management Schemes

Note that our proposed scheme can generate several pieces of secret information for a piece of ID. In this section, we show two systems which apply this property.

5.5.1 Group Communication Using Conference Key

Conference key means a generalization of two-party key establishment to provide three or more parties with a shared key. Note that despite superficial resemblance, conference keying protocols differ from dynamic secret sharing schemes in fundamental aspects. General requirements for conference key include that distinct groups recover distinct keys (session keys).

Figure 5.2 shows a example which apply our proposed scheme. In this case, we assume:

1. There exist three groups X, Y and Z.
2. Each group has several users.
3. Each user want to communicate with all the users which belong to a group.

In this case, the center generates several pieces of user's secret information S_{li} ($l = X, Y, Z$ and $i = A, B, C, \ldots$) which contain "group ID"-based signatures. Next, the center delivers it for each user. Note that each user's secret information has distinct value.

If user A want to communicate with all the users which belong to group Y, A only to makes a conference key by using his/her secret information s_{XA}. This system is profitable because one user can communicate with plural users.

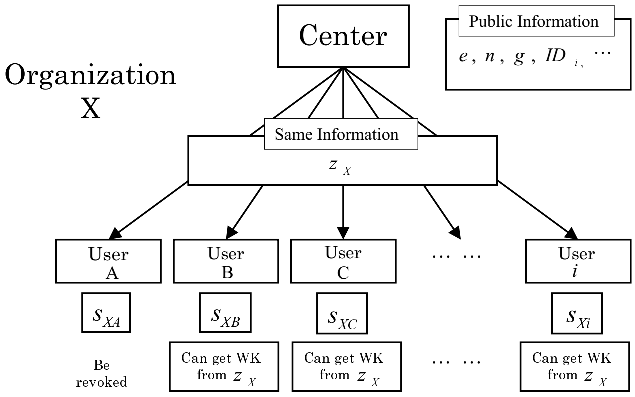


Figure 5.3: Example of broadcast communication system.

5.5.2 Revocation of Specific User Using Broadcast Communication

When a authority enables users of a (typically large) privileged subset to share a key by broadcasting one or more messages, the process resembles pre-positioned secret sharing and is called *Broadcast Communication.*

Figure 5.3 shows a example which apply our proposed scheme. In this case, we assume:

1. There exists a organization X which contains a center and plural users.
2. Each user already has a secret information $s_{Xi}(i = A, B, C, \ldots)$ and a common key K which are generated by the center.
3. s_{Xi} contains "organization ID"-based authentication. Each s_{Xi} has distinct value.
4. The center want to revoke a common key K and renew a new one to all members except for a specified user A.

In this case, the center broadcast a same information $Z_X$which can revoke only A's common key.Note that each user's secret information s_{Xi} contains a "organization ID"-based authentication and each information is distinct value. By using this difference, the center enable all users except for A to give a new common key.

If this system realizes as above, it is profitable because no efficient system is proposed until now.

Chapter 6

Conclusion

Today, network security is at unprecedented risk. Although Internet has enabled us to communicate with each other on networks which reach around the world, it caused some problems such as wiretapping, forgery, impersonation, and so on. Unfortunately these problems have getting terribly serious.

As the motivation for this work, we want to propose a practical scheme which gives the user less burden and more secure environment. Such schemes are necessary to realize a secure communication and establish the ideal infrastructures for network communication.

The major contribution of this thesis is to propose a new concept of identity-based cryptosystem which we call this system "Identity-based fault-tolerant key distribution system" and have showed an actual scheme by modifying the Okamoto-Tanaka key exchange scheme [8].

This system can generate several pieces of secret information for one ID. This means that it keeps one-to-one correspondence between users and ID's. This property enables the center to generate a new secret information even after the center has revoked one.

Next, we have studied the security of our proposed scheme using the reduction among functions. We have shown the breaking of our proposed scheme is equivalent to breaking the Diffie-Hellman key exchange scheme, and the security against impersonation depends on the difficulty of computing the RSA encryption scheme. This is the same as original one.

Finally, we have considered the application of the proposed concept to expand it into other key management.

Bibliography

[1] W. Diffie and M. E. Hellman, "New directions in cryptography", *IEEE Trans. Information Theory*, vol. IT-22, pp.644-654, 1976.

[2] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystem", *Communications of ACM*, vol. 21, pp.120-126, 1978.

[3] A. Shamir, "Identity-based cryptosystems and signature schemes", *Advances in Cryptology - CRYPTO'84*, LNCS 196, Springer-Verlag, pp.47-53, 1985.

[4] M. Tada, "Computational Complexity Theory for Cryptology", *private paper* in Japanese, 1998.

[5] E. Okamoto, "An Introduction to the Theory of Cryptography", Kyoritsu Shuppan, in Japanese, 1993.

[6] D. R. Stinson, CRYPTOGRAPHY Theory and Practice, CRC Press, 1995.

[7] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, HANDBOOK of APPLIED CRYPTOGRAPHY, CRC Press, 1996.

[8] E. Okamoto and K. Tanaka, "Key distribution system based on identification information", *IEEE J. Selected Areas in Communications*, Vol.7, pp.481-485, 1989.

[9] S. Tsujii, T. Itoh and K. Kurosawa, "ID-based cryptosystems using a discrete logarithm problem", *Electronics Letters*, pp.1318-1320, 1987.

[10] T. Matsumoto and H. Imai, "On the key distribution system: A practical solution to the key distribution problem", *Advances in Cryptology - Proceedings of CRYPTO'87*, LNCS 293, Springer-Verlag, pp.185-193, 1987.

[11] H. Tanaka, "Identity-Based non-interactive key sharing", *IEICE Trans. on Fundamentals of Electronics, Communication and Computer Science*, vol.E-77-A, no.1, pp.20-23, January 1994.

[12] H. Sakazaki, E. Okamoto and M. Mambo, "The Application of ID-Based Key Distribution Systems to an Elliptic Curve", *Proceedings of Information Security Workshop'97*, LNCS 1396, Springer-Verlag, pp.335-344, 1997.

[13] H. Tanaka, "Identity-Based Non-Interactive Key Sharing to RSA Public-Key Cryptsystem", *The 1998 Symposium on Cryptography and Information Security*, SCIS'98, 1998.

[14] N. Matsuzaki and J. Anzai, "A Group Key Renewal Method Suitable for Mobile Telecommunications(I)", *The 1998 Symposium on Cryptography and Information Security*, SCIS'98, 1998.

[15] T. Okamoto, M. Tada and E. Okamoto, "Identity-Based Fault-Tolerant Key Distribution System", *The 1999 Symposium on Cryptography and Information Security*, SCIS'99, 1999.

[16] K. Sakurai and H. Shizuya, "Relationships among the computational powers of breaking discrete log cryptosystems", *Advances in Cryptology - Proceedings of EUROCRYPT'95*, LNCS 921, Springer-Verlag, pp.341-355, 1995.

[17] M. Mambo and H. Shizuya, "A Note on the Complexity of Breaking Okamoto-Tanaka ID-Based Key Exchange Scheme", *Proceedings of Public Key Cryptography'98*, LNCS 1431, Springer-Verlag, pp.258-262, 1998.

[18] K. S. McCurley, "A key distribution system equivalent to factoring", *Journal of Cryptology*, No.1, Springer-Verlag, pp.95-105, 1988.

[19] M. Girault, "An identity-based identification scheme based on discrete logarithms modulo a composite number", *Advances in Cryptology - Proceedings of EUROCRYPT'90*, LNCS 473, Springer-Verlag, pp.481-486, 1990.

[20] M. Girault, "Self-certified public keys", *Advances in Cryptology - Proceedings of EUROCRYPT'91*, LNCS 1403, Springer-Verlag, pp.308-318, 1998.

[21] T. Okamoto and S. Uchiyama, "A New Public-Key Cryptosystem as Secure as Factoring", *Advances in Cryptology - Proceedings of EUROCRYPT'91*, LNCS 547, Springer-Verlag, pp.490-497, 1991.

[22] E. Bach, "Discrete logarithms and factoring", *Technical Report UCB/CSD 84/186*, University of California, Computer Science Division (EECS), 1994.

[23] H. Woll, "Reductions among number theoretic problems", *Information and Computation*, Vol.72, pp.167-179, 1987.

List of Publications

[1] T. Okamoto, M. Tada, E. Okamoto and K. Kamachi, "Revocable and Renewable ID-based Key Distribution System for User's Secret Information", *Proceedings of First Japan-Singapore Joint Workchop on Information Security,* JWIS'98 pp.3-13, 1998.

[2] T. Okamoto, M. Tada and E. Okamoto, "Identity-Based Fault-Tolerant Key Distribution System", *The 1999 Symposium on Cryptography and Information Security,* SCIS'99 pp.153-158, 1999.