공평한 비밀정보 교환

이 용 주[†] · 최 영 일[††] · 이 병 선[†††]

요 약

공평한 비밀정보 교환이란 서로 신뢰하지 못하는 당사자간에 공평한 비밀정보 교환의 목적을 가지는 프로토콜을 말한다. 두 당사자가 비밀정보를 교환할 때, 어느 한 쪽이 프로토콜을 임의로 중지하거나 부정을 저지름으로 인하여 다른 한 쪽이 불이익을 당하지 않는 특징을 가지며, 서로 신뢰하지 못하는 당사자간의 거래에 있어서 공평한 비밀정보 교환은 가장 중요한 요소가 된다. 이 논문에서는 새로운 공평한 비밀정보 교환을 설계하기 위해 기본개념과 정의에 대해 살펴보고, ElGamal의 공개키를 이용한 비대화형 OT 프로토콜을 설계한 후 설계한 프로토콜을 이용해 새로운 공평한 비밀정보 교환 알고리즘을 설계한다. 새로운 공평한 비밀정보 교환에서는 서로 신뢰하지 못하는 두 당사자가 비밀 정보를 불확정 전송하며 자신이 수신한 정보가 참인지를 양방향으로 확장한 영지식 증명을 이용하여 검증한다. 이때 두 당사자는 상대의 비밀정보를 복호화 할 수 없고, 양측의 검증 절차가 끝난 후 복호화가 가능하다. 상대의 신원을 밝히지 않아도 부정을 저지를 수 없는 특성 때문에 익명성과 공평성을 제공한다는 것이 가장 큰 장점이다.

How to Exchange Secrets by OT

Yongju Yi[†] · Young-Il Choi[††] · Byung-Sun Lee[†††]

ABSTRACT

A fair exchange protocol enable two parties to exchange secrets with fairness, so that neither can gain any information advantage by quitting prematurely or otherwise misbehaving. Therefore a fair exchange is the most important for electronic transactions between untrusted parties. To design new fair exchange, after describing basic concepts, definitions and existing protocols and designing a non-interactive OT protocol using ElGamal's public key system, I will design new protocol to support fair exchange. In my designed new protocol, untrusted parties exchange secrets obliviously and verify that their received secrets are true by using transformed Zero Knowledge Interactive Proof extended to duplex. At this time, concerned two parties can't decrypt the other's ciphertext. After all of the steps, two parties can do it. It is the most important to provide perfect fairness and anonymity to untrusted parties in this protocol.

키워드 : 불확정전송(OT), 영지식증명(ZKIP), 공평한 비밀정보 교환(Fair Exchange), 엘가말(ElGamal)

1. Introduction

1.1 objective

As the supply of computer and use of internet are increased, the amount of information via computer network is on the sudden increase. The increase of amount of information cause the problem of authentication and stability like interruption, interception, fabrication, masquerade, replay, denial of service and modification for lawful users. Therefore more detailed requirements for security and expansion for authentication market are needed.

One of the big issues for information protection is the problem about secrecy of information and authenticity for lawful user. The method of protecting secrets from wrong of legal user as well as third party is needed. Countermeasure to prevent concerned parties's wrong than third party's wrong is more needed as the form of electronic commerce has been changed variously in the next generation internet.

In this environment, when parties exchange secrets, fairness, concurrency and anonymity become security requirements which are to prevent parties from getting disadvantage. Therefore in this paper, We describe Tom Tedrick's fair exchange as the most typical method among existing fair exchanges, then analyze problems of his protocol and design new protocol. We apply oblivious transfer protocol as basic tool providing fair exchange and transformed zero knowledge interactive proof to remove disadvantage which one of two parties may have to our new fair exchange. The security problem described in this paper is not only to third party but also to concerned two parties. We design protocol in which two parties entrust minimum information to Trusted Server (TS) to prevent the other party's wrong. If one of two parties does wrong before ending protocol, the party who halts protocol can't have any information or computational advantage. If one of two parties do wrong after ending protocol, the other party can ask TS about the other party's secrets.

† 정 회 원 : 한국전자통신연구원 연구원
†† 정 회 원 : 한국전자통신연구원 책임연구원 소프트스위치팀
††† 정 회 원 : 한국전자통신연구원 책임연구원 소프트스위치 팀장
논문접수 : 2002년 7월 12일, 심사완료 : 2002년 10월 24일

1.2 The introduction of fair exchange

A fair exchange protocol enables two parties to exchange secrets with fairness, so that neither can gain any information advantage by quitting prematurely or otherwise misbehaving [2]. Therefore a fair exchange is the most important for electronic transactions between untrusted two parties.

A new fair exchange has been proposed to meet the qualifications of a fair exchange and to offer anonymity. It ensure that if you pay for an item, then you will receive the goods. Our new fair exchange protocol use a trusted server as a trusted third party to facilitate the exchange.

2. Background

2.1 Definitions

In this section, We introduce definitions used in this paper.

2.1.1 The RSA Problem

The intractability of the RSA problem forms the basis security of the RSA public-key encryption and the RSA signature scheme.

The *RSA Problem* (RSAP) is the following : given a positive integer n that is a product of two distinct odd primes p and q, a positive integer e such that $gcd\ (e, (p-1)(q-1)) = 1$, and an integer c, find an integer m such that $m^e \equiv c (mod\ n)$.

In other words, the RSA problem is that of finding e^{th} roots modulo a composite integer n. The conditions imposed on the problem parameters n and e ensure that for each integer $c \in \{0, 1, \cdots, n-1\}$ there is exactly one $m = \{0, 1, \cdots, n-1\}$ such that $m^e \equiv c \pmod n$. Equivalently, the function $f: Z_n \rightarrow Z_n$ defined as $f(m) = m^e \bmod n$ is a permutation.

2.1.2 The Discrete Logarithm Problem

The security of many cryptographic techniques depend on the intractability of the discrete logarithm problem. Partial lists of these include Diffie-Hellman key agreement and its derivatives, ElGamal encryption, and the ElGamal signature scheme and its variants [3]. Let G be a finite cyclic group of order n. Let a be a generator of G, and let $\beta \in G$. The discrete logarithm of β to the base a, denoted $\log_a \beta$, is the unique integer x, $0 \le x \le n-1$, such that $\beta = a^x$.

The *generalized discrete logarithm problem* (GDLP) is the following : given a finite cyclic group G of order n, a generator g of G, and an element $B \in$ G, find the integer x, $0 \le x \le n-1$, such that $g^x = B$.

2.1.3 The Diffie-Hellman Problem

The Diffie-Hellman problem is closely related to the well-studied discrete logarithm problem (DLP). It is of significance to public-key cryptography because its apparent intractability form the basis for the security of many cryptographic schemes including Diffie-Hellman key agreement and its derivatives, and ElGamal public-key encryption.

The Diffie-Hellman Problem (DHP) is the following : given a prime p, a generator a of Z_p^*, and elements $a^a \bmod p$ and $a^b \bmod p$, find $a^{ab} \bmod p$.

The generalized Diffie-Hellman (GDHP) is the following : given a finite cyclic group G, a generator a of G, and group elements a^a and, a^b fine a^{ab}.

Suppose that the discrete logarithm problem in Z_p^* could be efficiently solved. Then given a, p, $a^a \bmod p$ and $a^b \bmod p$, one could first find a from a, p and $a^a \bmod p$ by solving a discrete logarithm problem, and then compute $(a^b)^a = a^{ab} \bmod p$. This establishes the following relation between the Diffie-Hellman Problem and the discrete logarithm problem.

2.2 Background

2.2.1 Oblivious Transfer

OT protocol provide fairness between untrusted two parties in the 1-out-of-2 OT protocol, when one party transmit secrets to the other. Therefore, in the first part of this chapter, we describe the concepts of OT protocol. You can use general term like information protection protocols as a countermeasure of secret data and justifiable users in cryptography and communication media. The protocol accepting cryptographic technology can be defined as the secure protocol.

Problem for fairness of secure requirements of secure protocol is very important. Fair exchange is the protocol in which men exchange fairly their own secret data [4].

For example, there are many problems when one of untrusted two parties sign a contract before in the internet. To solve these problems, the research on oblivious transfer for fair exchange is required.

OT protocol is sub-protocol used as a basic implement to design secure protocol. OT protocol can be usefully used in specific secure protocol protecting the security of data and sending related secure data at the same time. When one party transmit secrets to the other, the latter has a chance of 1/2 of obtaining secrets and the former can guess whether receiver receive the secrets with probability one half [5].

Of course, if receiver has a chance of success, receiver can decrypt the ciphertext. The most important feature in the OT protocol is that sender and receiver can't misbehave each other since all chances is an exact 1/2.

2.2.2 Zero Knowledge Interactive Proof

In 1985, Coldwasser, Micali and Rodkoff introduced zero knowledge interactive proof [ZKIP]. The idea of ZKIP is innovative for authenticating the other party in that a verifier don't open any information excepting information about the reasonability of proof. From those day forth many theories for ZKIP have been published and methods of these theories authenticate the other party by using languages belong to *NP*. Coldwasser, Micali and Rodkoff prove that all languages belong to *NP* can used in all of the ZKIP.

ZKIP is the method testifying justifiability between a proofer (P) and a verifier (V) through interaction and has meaning of exchanging information about justifiability of the event. Exactly, Proofer (P) can prove that he is the only man who know secrets to a verifier (V) by transmitting other information instead his secrets.

ZKIP is method generalizing the *NP* style. The *NP* style is made up of two type of decision turing machines. One is a proofer (P) which has infinite computational capability, the other is a verifier which has computational capability for polynomial. A proofer and a verifier receive *NP* problem (X) as a common input. If a proofer which has infinite computational ability get the solution (a) of a problem (X) and send this value to a verifier in the *NP* proof method, a verifier which has a computational capability for polynomial decides that a is the solution of a problem (X). A proofer (P) computes the solution (a) of a problem (X) by receiving a input (X) and sends the result to a verifier through a communication tape. A verifier (V) authenticates a proofer by testifying the solution (a) of a problem (X) which is received from a proofer.

The most general example of the *NP* proof method is user authentication using password. When a system user (P) transmit password to a computer system (V), a computer system (V) authenticates that the owner of this password is a justifiable user. The applicable fields of the *NP* proof method in cryptography are very limitable because a which a proofer send to a verifier is very important. Therefore to overcome this weakness, new proof methods are needed.

3. RELATED WORKS

3.1 Related works

We consider several problems which arose in the context of "the exchange of Secret Keys," [4] in which one party may halt the exchange protocol and has a 2 to 1 expected time advantage in computing the other party's secrets. To solve this problem, when there is a particular point in the exchange where this time advantage may be critical, Tom Tedrick presented a method for exchanging "fraction" of a single bit at CRYPTO 83 [15]. He also extended the method so as to apply it to all bits to be exchanged at CRYPTO 84 [16]. He decreased the maximum computational advantage and provided more powerful fairness. But there is any unfairness and burden to computers and communication media yet. To solve these problems, Berger and Tedrick proposed a fair exchange using OT protocol at EUROCRYPT 84 [17]. But one of two parties may have any disadvantage because all of the two parties have a chance of 1/2 of obtaining messages [18]. To solve these problems, We use OT Protocol and transformed ZKIP. In the following sections, we introduce fair exchanges of Blum and Tom Tedrick.

3.2 How to Exchange Half a Bit

Blum proposed a method to exchange secrets between untrusted two parties. In the exchange of secrets, two adversaries (Client A and Client B) have composites $Na = Pa \times Qa$, where Pa, Qa, Pb and Qb (the keys) are large primes congruent to 3 mod 4. Client A knows the factorization of Na, Client B knows the factorization of Nb. They use following protocol in order to exchange their secret keys. Client A selects 100 integers X1, X2, X100 $<$ Nb at random, squares them Mod Nb, and sends the squares to Client B. (Client B then sends back both distinct roots of each Xi^2, a bit at a time, in exchange for similar information from Client A. When Client A knows both distinct roots of any Xi^2 she can factor Nb. In order to completely cheat Client A by sending bad information Client B must guess which root of Xi^2 is known to her in all 100 cases, hence chance of success is $1/2^{100}$. But there is a problem. Suppose Client A goes first. If she sends a bit at a certain point in the protocol, and Client B does not respond, he may know 1 more bit than she does. He may then expect to factor Na in half the time it takes Client A to factor Nb. To solve this problem, Tom Tedrick proposed solution.

The solution is that instead of sending a bit of a root Client A can send a "fraction" of a bit as follows. She simply says, for example, the next 3 bits are NOT 011. If client B stops the protocol, he will needs about 7/8 much time to factor Na as client A does to factor Nb. She can send almost arbitrarily small fractions of a bit in this way.

Theorem

Client B has exactly the same chance of completely cheating Client A as in the original protocol.

Proof

In order to cheat client A by sending bad information Client B must guess which root in each pair is known to Client

A, and claims certain bits of the other root are not what they really are. Since Client A knows half the roots, Client B has a 50% chance of getting caught each time he tries to cheat, hence his chance of completely cheating Client A is $1/2^{100}$, as before.

3.3 Fair exchange

Tom Tedrick proposed a method whereby two adversaries can exchange information worth an arbitrarily small "fraction of a bit" in a particular setting, although neither trusts the other in "How to Exchange Half a Bit".

Instead of sending a bit of a root, Client A can send a "fraction" of a bit as follows. She simply says, for example, the next 3 bits are NOT 011. If Non stops protocol, he will need about 7/8 as much time to factor *Na* as Client A does to factor *Nb*. She can send almost arbitrarily small fractions of a bit in this way.

But Client B has exactly the same chance of completely cheating Client A as in the original protocol. This is a big problem. In order to cheat Client A by sending bad information Client B must guess which root in each pair is known to Client A, and claim certain bits of the other root are not what they really are. Since Client A knows half the roots, Client B has a 50% chance of getting caught each time he tries to cheat, hence his chance of completely cheating Client A is $1/2^{100}$, as before.

He extended the method so as to apply it to all bits to be exchanged in "Fair Exchange of secrets" to solve this problem himself.

4. A Non-interactive OT Protocol using ElGamal's public key Encryption

We designed a non-interactive OT protocol for Fair Exchange Technique (FE) as one of encryption protocols to reduce computing time and communication bits which are the problems in a interactive protocol in this chapter. We use a ElGamal's public key encryption for this protocol which has non-repudiation [19].

4.1 Execution steps of protocol

ElGamal published ElGamal's public key encryption system using the key distribution method based on the discrete logarithm problem of Diffie-Hellman [20].

When a sender transmit $(M)(1 \le M \le p-1)$ to receiver, the protocol is processed separately in key generation, encryption and decryption and processing step is like following.

- **Key generation step**

① Client A generates a large random prime p and a generator g of the multiplicative group Z_p^* of the integers modulo p, selects random variable $(x)(1 \le x \le p-1)$ and computes $y = g^x \pmod p$. He sends results (p, y, g) to trusted server (TS) and keeps private key (x) secretly.

② TS stores the value (p, y, g) received from Client A and opens this results to the public.

- **Encryption**

③ Client B receives Client A's public keys (p, y, g) from TS. He selects message $(M)(M \in \{1, 2, \cdots, p-1\})$ and $(k)(1 \le k \le p-1)$, computes $W = g^x \pmod p$ and $Z = M \cdot y^x \pmod p$ and sends $(C) = (W, Z)$ to Client A.

- **Decryption**

④ Client A decrypt (M) from cipher text $(C) = (W, Z)$ by computing $Q = W^{-x} = \pmod p = g^{-xk} \pmod p$ using (x).

$$M = Q \cdot Z = g^{-xk} \cdot (M \cdot y^k) \pmod p$$
$$= g^{-xk} \cdot Mg^{xk} \pmod p = g^{-xk} \cdot Mg^{xk} \pmod p = M$$

The processing step of non-interactive oblivious transfer protocol using ElGamal's public key encryption system designed in this chapter is like following.

Notation

- x_a : Client A's private key
- x_b : Client B's private key
- a_0, a_1 : Client A's secrets
- p, g, A_i, A_{1-i} : Client A's public data
- S_0, S_1 : Secrets which Client A sends Client B

Registration

① Client A selects large primes $(p, g \in Z_p^*)$ and private key $(x_a \in \{1, 2, \cdots, p-1\})$, computes $A_i = g^{x_a} \pmod p$, $A_{1-i} = (g^{x_a})^{-1} \pmod p$ and sends results (p, g, A_i, A_{1-i}) to TS.

② TS keeps (p, g, A_i, A_{1-i}) and opens these values to the public.

Transmission

③ Client B selects private key $(x_b \in \{1, 2, \cdots, p-1\})$ and encrypts secrets (S_i) like $(W_i = g^{x_b} \pmod p)$. $(i \in \{0, 1\})$. He computes $Z = S_i (A_i)^{x_b} \pmod p$, $Z = S_i (A_{i-1})^{x_b} \pmod p$ and sends results (W_i, Z) to Client A.

④ Client A obtains secret (S_i) by computing $Q = (W_i)^{-x_a} \pmod p$, $S_i = (W_i)^{-x_a} Z \pmod p$ from (W_i, Z) received from Client B.

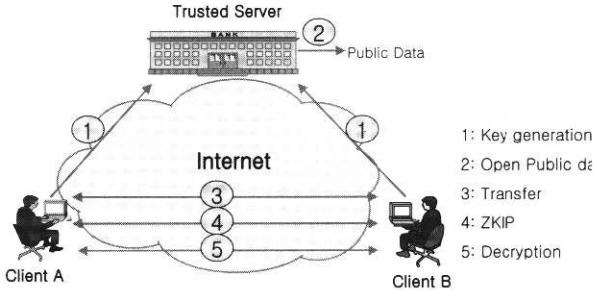
5. A New Fair Exchange

In this chapter, we will describe the fair exchange of secrets by oblivious transfer satisfying the following definition.

5.1 Definition and Condition in proposed protocol

We can say that fair exchange is really fair if concerned parties have the fairness, soundness and privacy.

Modeling Protocol : This protocol consists of key generation, transfer, transformed ZKIP and decryption.



(Figure 1) The model of Protocol

5.2 Notation and Assumption

Notation

- x_a, a : Client A's secret keys.
- x_b, b : Client B's secret keys.
- $p, g \in Z_p^*$: Client A and Client B's public data which Trusted Server opens to the public.
- $A_i, A_{1-i}, K_i, K_{1-i}$: Client A's public data.
- $B_j, B_{1-j}, T_j, T_{1-j}$: Client B's public data.
- S_a, i : secrets which Client A sends to Client B
- S_b, j : secrets which Client B sends to Client A

Assumption

We presume that next items are

① Two partie's nick names are Client A and Client B. Their identifications are disclosed perfectly.

② Trusted Server is a trusted server and generates keys. This server is trusted by two parties.

③ Communication channel between Client A and Client B is secure.

5.3 A more formal description of the method

Client A and Client B don't need to disclose their identities. Trusted Server functions as a key generator and stores one key secretly against one party's misbehaving. But two parties can't misbehave each other, because secrets are transmitted obliviously for two parties not to guess the result. The transmission of data is achieved one time. Then two parties verify received ciphertext using transformed ZKIP to confirm whether their secrets are true. At this time, two parties can't decrypt the secret. This step is to remove disadvantage any one party may have. After this step, If all of the two parties have secrets, inform Trusted Server of the result. After TS's receiving Y from two parties, Two parties send the last key to counterpart. Because Client B acts similarly in the every step, We write only symbols and computational formulas shortly.

5.3.1 Key generation

This step is to generate the key pair (public, private) of Trusted Server, Client A and Client B who are members of this protocol. All of the secret keys were kept by owner secretly.

① TS

generates randomly a large prime p and a generator g of the multiplicative group Z_p^* and modulo p, selects $e, d = 1 \pmod{p}$, opens T_e(public key) to the public and keeps T_d (TS's private key) secretly.

② Client A

After selecting an integer $i \in \{0, 1\}$, Client A keeps it secretly. Client A selects random integers $(x_a)(1 \le x_a < p-1)$ and $(a)(1 \le a < p-1)$ about public large random prime p, computes public data $A_i = g^{x_a} \pmod{p}$, $A_{1-i} = (g^{x_a})^{-1} \pmod{p}$ and $K_i = g^a \pmod{p}$, $K_{1-i} = (g^a)^{-1} \pmod{p}$, encrypts $P_a = T_e(a)$ with TS's public key (T_e), sends the ciphertext (A_i, A_{1-i}, K_i, K_{1-i}, P_a) to TS, and keeps the private keys (x_a, a) secretly.

③ TS

Trusted Server opens A_i, A_{1-i}, K_i, K_{1-i} among the values received from Client A to the public, recovers a by computing $a = T_d(T_e(a))$ and verifies a by computing $K_{1-i} \cdot K_i = g^a \cdot (g^a)^{-1}$. After all of the steps, Trusted Server keeps a secretly.

④ Client B

After selecting an integer $i \in \{0, 1\}$, Client B keeps it secretly. Client B selects random integers $(x_b)(1 \le x_b < p-1)$ and $(b)(1 \le b < p-1)$ about public large random prime p, computes public data $B_j = g^{x_b} \pmod{p}$, $B_{1-j} = (g^{x_b})^{-1} \pmod{p}$ and $T_j = g^b \pmod{p}$, $T_{1-j} = (g^b)^{-1} \pmod{p}$, encrypts $P_b = T_e(b)$ with TS's public key (T_e) sends the ciphertext ($B_j, B_{1-j}, T_j, T_{1-j}, P_b$)) to TS and keeps the private keys (x_b, b) secretly.

⑤ TS

Trusted Server opens $[B_j, B_{1-j}, T_j, T_{1-j}]$ among the values received from Client B to the public, recovers b by computing $b = T_d(T_e(b))$ and verifies b by computing $T_{1-j} \cdot T_j = g^b \cdot (g^b)^{-1}$ After all of the steps, Trusted Server keeps b secretly.

5.3.2 The transmission of secrets

⑥ Client A <==> Client B

<Client A>

Client A selects B_i among Client B's public data, computes $D_i = ((S_a \cdot a) \cdot (B_i)^{x_a} \bmod p)$ and sends D_i to Client B.

<Client B>

Client B selects A_j among Client A's public data, computes $D_j = ((S_b \cdot b) \cdot (A_j)^{x_b} \bmod p)$. and sends D_j to Client A.

5.3.3 Zero Knowledge Interactive Proof

Used ZKIP in this protocol is to verify that a party's secret bit (i or j) corresponds correctly to the other's. Two parties have a chance of 1/2 of obtaining secrets. If one party fails to obtain the secrets, the party halts the protocol promptly. In this step, All of the two parties can't decrypt the ciphertext. Therefore two parties don't have any reason to reject this verification. Though protocol is halted, Two parties don't have any exposure of their identities.

⑦ Client A <==> Client B

<Client A>

Client A selects a random integer $r_a \in Z_P$, computes $R_a = g^{r_a} (\bmod p)$ and sends the result to Client B.

<Client B>

Client B selects a random integer $r_b \in Z_P$, computes $R_b = g^{r_b} (\bmod p)$ and sends the result to Client A.

⑧ Client A <==> Client B

<Client A>

After receiving R_b from Client B, Client A selects $ea \in \{0,1\}$ randomly and sends it to Client B.

<Client B>

After receiving R_a from Client A, Client B selects $eb \in \{0,1\}$ randomly and sends it to Client A.

⑨ Client A <==> Client B

<Client A>

After receiving e_b from Client B, if e_b is 0, sends $Z_a = r_a (\bmod p)$ to Client B, or e_b = 1, computes $Z_a = r_a + a (\bmod p)$ and sends it to Client B.

<Client B>

After receiving e_a from Client A, if e_a is 0, sends $Z_b = r_b (\bmod p)$ to Client A, or e_a = 1, computes $Z_b = r_b + b (\bmod p)$ and sends it to Client A.

⑩ Client A <==> Client B

<Client A>

After receiving Z_b from Client B, if e_a is 0, computes $R_b = g^{Z_b} (\bmod p)$. or e_a = 1, computes $R_b = g^{Z_b} \cdot T_{(1-i)} (\bmod p)$ and verifies R_b. If the computational formula isn't true, Client A halts protocol for not having secrets. If e_a = 0, It is insecure for Client A to obtain secrets. If Client A chooses ea at perfect random and repeats n times, the probability of her choosing 1 as e_a is the following formula because she has a chance 1/2 of obtaining 1, $\frac{1}{2} + (\frac{1}{2})^2 + \cdots + (\frac{1}{2})^n = 1 - (\frac{1}{2})^n$. If n is large enough, she can check in all probability.

<Client B>

After receiving Z_a from Client A, if e_b is 0, computes $R_a = g^{Z_a} (\bmod p)$. or e_b = 1, computes $R_a = g^{Z_a} \cdot K_{(1-j)} (\bmod p)$ and verifies Z_a. If the computational formula isn't true, Client B halts protocol for not having secrets. If e_b = 0, It is insecure for Client B to obtain secrets. If Client B chooses e_b at perfect random and repeats n times, the probability of his choosing 1 as e_b is the following formula because he has a chance 1/2 of obtaining 1, $\frac{1}{2} + (\frac{1}{2})^2 + \cdots + (\frac{1}{2})^n = 1 - (\frac{1}{2})^n$. If n is large enough, He can check in all probability.

⑪ Client A <==> Client B

Client A and Client B inform Trusted Server of the result Y or N.

5.3.4 Decryption

As Client A's secret data i is identical with Client B's secret j, Two parties transmit one decryption key and decrease the amount of transmission by using the other key required in public data.

⑫ Client A <==> Client B

<Client A> Client A sends a to Client B.

<Client B> Client B sends b to Client A. After computing $Q_j = (A_j)^{-x_b} (\bmod p)$, Client B computes $Y_j = Q_j \cdot D_i (\bmod p) = S_a \cdot a (\bmod p)$ and decrypts $S_a = \frac{S_a \cdot a}{a}$ using a received from Client A.

⑬ Client A <==> Client B

<Client A>

After computing $Q_i = (B_i)^{-x_a} (\bmod p)$, computes $Y_i = Q_i$

$\cdot D_j(\bmod p) = S_b \cdot b(\bmod p)$ and decrypts $S_b = \dfrac{S_b \cdot b}{b}$ using b received from Client B.

If Client B halts protocol after receiving Client A's decryption key in [step 11], Client A sends $P_a = T_e(a)$ to TS. After confirming Y received from two parties, Trusted Server send P_b to Client A.

6. Analysis of Protocol

6.1 The Analysis of NIOT using ElGamal's Public Key Encryption

When designed protocol in this paper is executed, it has two cases. The one is that Client A knows Client B's Secrets (S_A) accurately. The other is that Client A has no any information about Client B's secrets (S_A). The probability of occurring two cases is exactly half. Client A and Client B can't know each other's private key because this protocol based on Diffie-Hellman problem which is that party having $g^x$and g^y can't compute g^{xy} without x and y. Therefore two parties can't do wrong. If one of two does wrong, the other party can detect the other party's cheat. This protocol is designed to remove the probability of conspiracy of TS and third party since private keys of Client A and Client B are not known to TS.

The table below describe the differences between general oblivious interactive transfer, general oblivious non-interactive transfer and non-interactive oblivious using ElGamal public key encryption.

〈Table 1〉 Comparison of IOT, NIOT and NIOT using ElGamal public key encryption

Contents of comparison	IOT	NIOT	NIOT using ElGamal PKE
Probability of an exposure of message during executing a protocol	×	×	×
Probability of two partie's doing wrong	○	○	○
Probability of detecting the other's wrong	×	×	○
Does TS know Two partie's private keys	×	×	×
Non-repudiation for sending message to the other later	×	×	○
Non-repudiation for receiving message to the other later	×	×	×
The amount of message to communicate and compute	○	△	△

× : non-supported, △ : weakly supported, ○ : well supported

6.2 The Analysis of New Fair Exchange

The protocol designed in CHAPTER 5 has four cases like following.

① Both Client A and Client B have a chance of success.
② Client A has a chance of success and Client B can't have it.
③ Client B has a chance of success and Client A can't have it.
④ Client A and Client B can't have a chance of success.

In case ②, case ③, case ④, Both client A and client B can't decrypt the other party's secrets. Two partie's identities are not exposed. In case ④, client A and client B can exchange secrets obliviously.

6.3 Stability and Fairness

If two parties follow this protocol fully, the probability of obtaining secrets is exactly 50%. That everything is 50% is a special feature of this protocol. Because one party check that his received secrets is true and let the other party confirm secrets simultaneously, that one party halt protocol at any step has not advantage or disadvantage. Note stability at each step. in key generation step, there can't be any wrong because of Trusted server's verification. In transmission of secrets, Two parties transmit secrets fairly to challenge to 50% probability. In transformed ZKIP, they check that their selected secrets is true simultaneously. Although one party's secrets is true, if the other party's secrets is false, he can't receive the last decryption key from the other party failed from 50%. Therefore two party all don't have any reason to reject this check step.

If Both choose e_a or e_b at perfect random and repeats n times, the probability of his choosing 1 as e_a or e_b is the following formula because They have a chance 1/2 of obtaining 1, $\frac{1}{2} + (\frac{1}{2})^2 + \cdots + (\frac{1}{2})^n = 1 - (\frac{1}{2})^n$. If n is large enough, they can check in all probability.

Now two party come decryption step, As Client A's secret data i is identical with Client B's secret j, two parties transmit one decryption key and decrease the amount of transmission by using the other key required in public data. Therefore two parties have fair exchange.

6.4 Problems of A New Fair Exchange

In proposed new fair exchange, we can point out these problems.

① This protocol is between two parties. Therefore fair exchange among multi parties is impossible.
② There should be a TS, that is, only two parties can't exchange secret with fairness without TS.
③ If one of two parties fail from 50%, their exchange is invalid.

7. Conclusions and Suggestions for Further Research

In this paper, we described basic concepts, definitions and existing protocols, designed a non-interactive OT protocol using ElGamal's public key system and proposed new fair exchange protocol to support fair exchange. In new protocol, untrusted two parties exchange secrets obliviously and verify that their received secrets are true by using transformed ZKIP extended to duplex.

Note that untrusted two parties have fairness and anonymity by using the 1-out-of-2 OT protocol and ZKIP. Also neither can gain an information advantage by quitting prematurely or otherwise misbehaving. This method is much more effective in that two parties transmit the ciphertext obliviously at one time than a method for exchanging fractions of bits. But detailed research is needed for the critical analysis of protocol and extension to multi parties.

References

[1] H. S. Shin, "Stability of E-Commerce and Authentication Service," Korea Information Processing Society Review, pp. 107-114, March, 2000.

[2] M. K. Franklin and M. K. Reiter, "Fair exchange with a semi-TTP," In Proceedings of the 4th ACM Conference on Computer and Communications Security, pp.1-6, April, 1997.

[3] Alfred J. Menezes, Paul C.van Oorschot, Scott A. Vanstone, "Handbook of APPLIED CRYPTOGRAPHY," 1997.

[4] M. Blum, "How to Exchange Secret Keys," ACM Trans. Compute System, pp.175-193, May, 1983.

[5] M. Blum, "Coin Flipping by Telephone," IEEE, COMPCON, pp.133-137, September, 1982.

[6] Halpern, J, Rabin, M., "A Logic to Reason about Likelihood," ACM Symposium on Theory of Computing, pp.10-319, May 1983.

[7] S. Even, O. Goldreich, A. Lempel, "A Randomized Protocol for Signing Contracts," Communications of the ACM, Vol.28, pp.637-647, 1985(Early Version : Proceedings of CRYPTO '82, Springer-Verlag, pp.205-210, 1983).

[8] M. Bellare, S. Micali, "Non-Interactive Oblivious Transfer and Applications," Advances in Cryptology : CRYPTO '89, pp.547-557, 1989.

[9] Santis, A, Persiano, G, "Public-Key Cryptography," Proceedings of EUROCRYPTO '90, May, 1990.

[10] C. Crepeau, "Quantum Oblivious Transfer," Journal of Modern Optics, Vol.41, No.12, pp.2445-2454, December, 1994.

[11] C. Crepeau, J. Graaf, A. Tapp, "Committed Oblivious Transfer and Private multi-party Computation," Advances in Cryptography Proceedings of CRYPTO '95, pp.110-123, 1995.

[12] C. K. Kang, D. Y. Kim, "Digital multiple signature in electronic contract signature System," EC, 1993.

[13] S. C. Kim, Y. S. Oh, S. H. Lee, "Oblivious Transfer with non-repudiation", KIISC Journal(A), No.26, Vol.3, pp.333-340, Mar, 1999.

[14] S. Goldwasser, S. Micali, C. Rackoff, "Knowledge Complexity of Interactive Proofs," Proc. 17th STOC, pp.291-304, 1985.

[15] T. Tedrick, "How to Exchange Half a Bit," Proceedings of CRYPTO '83, pp.147-151, 1983.

[16] T. Tedrick, "Fair Exsnange of Secrets," Proceedings of CR YPTO '84, pp.434-438, 1984.

[17] R. Peralta, R. Berger, T. Tedrick, "A Provably Secure Oblivious Transfer," Proceedings of EUROCRYPT '84, pp. 379-386, 1984.

[18] Rabin, M., "How to Exchange Secret by Oblivious Transfer," Harvard Center for Research in Computer Technology, Cambridge '81, 1981.

[19] S. C. Kim, Y. J. Yi, S. H. Lee, "Non-interantive Oblivious Transfer Protocol Using ElGamal Public Key Encryption," Proc. kiisc-cc, pp.15-26, December, 1999.

[20] T. ElGamal, "A Public Key Cryptosystem and Signature Scheme Based on Discrete Logarithms," IEEE Trans. Information Theory, Vol.31, No.44, pp.469-472, 1985.

이 용 주

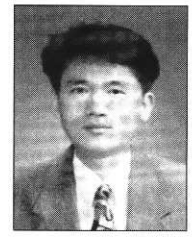
e-mail : silvia@etri.re.kr

1999년 청주대학교 정보통신공학과(학사)

2001년 충북대학교 전산학과(석사)

2001년~현재 한국전자통신연구원 연구원

관심분야 : 개방형서비스기술, 개방형 프로토콜, 미들웨어



최 영 일

e-mail : yichoi@etri.re.kr

1983년 서울대학교 전자공학과(공학사)

1998년 충남대학교 컴퓨터과학과(이학석사)

2002년 충남대학교 컴퓨터과학과(이학박사)

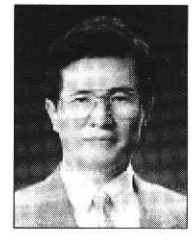
1996년 정보통신 기술사

1985년~1986년 AT&T Bell 연구소 객원연구원

2002년 표준화 전문위원(MSF, Parlay)

1983년~현재 한국전자통신연구원 책임연구원(소프트스위치 팀)

관심분야 : 차세대 네트워크, Softswitch, 개방형 서비스 기술



이 병 선

e-mail : bslee@etri.re.kr

2002년 한국과학기술원 박사(전산학)

1982년~2003년 한국전자통신연구원 책임연구원(TDX 계열 교환기, ATM 교환기, 소프트스위치 개발)

현재 소프트스위치 팀장

현재 MSF 이사, NGcN 포럼 통합망 분과위원장

관심분야 : Real-time Software, Software Reliability, Fault-Tolerant Computing, Model-Driven Architecture