대칭형 인증 및 키 교환 프로토콜을 이용한 비대칭형 프로토콜의 설계 기법

(Method to Obtain Asymmetric Authenticated Key Exchange Protocols from Symmetric Ones)

양 대 헌 †

(DaeHun Nyang)

요 약 대칭형 패스워드 기반의 인증 및 키교환 프로토콜은 비대칭형 프로토콜보다 설계, 분석, 효율면에서 더 좋은 성질을 가진다. 하지만, 인증 서버가 공격당하는 경우 대칭형 인증 프로토콜은 쉽게 사용자의 패스워드가 노출된다. 비대칭형의 안전성 증명을 가지는 PAK-X나 SNAPI-X같은 프로토콜이 제안되었지만, Diffie-Hellman 키교환에 비해 많은 연산을 필요로 하고 있다. 이 논문에서는 패스워드 기반의 대칭형 인증 및 키교환 프로토콜을 비대칭형 프로토콜로 변환하는 효율적인 방법을 제시한다.

키워드 : 암호, 인증, 키교환, 패스워드, 영지식, 비대칭 모델

Abstract Password authenticated key exchange protocols for the symmetric model are easier to design, analyze and are more efficient than ones for the asymmetric model, but they are most likely to be broken by server's compromise. Though the protocols with provable security for the asymmetric model such as PAK-X and SNAPI-X are introduced, they need large amount of computation compared with the standard Diffie-Hellman key exchange. We present a systematic and efficient way to transform password authenticated key exchange protocols for the symmetric model into protocols for the asymmetric model. Thus, an efficient protocol for the asymmetric model can be constructed by a systematic protocol with low computation.

Key words : Authenticated Key Exchange, Password, ZKIP, Asymmetric Model

1. Introduction

Though public key cryptography is now very popular and a powerful tool in cryptography, it is inconvenient to carry a public/private key pair. In some application, a password is enough to prove a user's identity. Password based protocol alleviates user's needs to carry a complex proving device such as smart cards. However, because a password must be easy for human to remember, it has low entropy, and it is short.

In 1992, Belloving and Merritt invented a new authentication protocol(EKE: Encrypted Key Exchange) that has an interesting feature[1]. Their protocol protects a user's password from the off-line guessing attack. After the introduction of EKE, SPEKE(Simple Password Exponential Key Exchange) by Jablon, has been followed[2]. The protocols integratethe session key exchange with password authentication. Results of augmenting those schemes into asymmetric model are B-SPEKE and A-EKE[4][3]. SRP(Secure Remote Password protocol), another password based authentication and key exchange protocol of the asymmetric model is designed by Tom Wu[5]. These protocols, however, do not have security proof.

The protocol of the asymmetric model gives a server a *password verifier* that is computed from a password but cannot be directly used to find out the password, whereas in the symmetric model both a server and a user have the same password. Protocols for the symmetric model is easier to

† 정 회 원 : 인하대학교 정보통신대학원 교수
nyang@inha.ac.kr
논문접수 : 2003년 9월 26일
심사완료 : 2003년 11월 27일

design, analyze and are more efficient than ones for the asymmetric model, but it is most likely to be broken by server's compromise.

Recently, Bellare, Pointcheval and Rogaway proves correctness for the idea at the center of the EKE[6] in the ideal cipher model. Their formal validation of security is the basis of our work. Also, authentication protocols with security proof in the random oracle model such as PPK(Password Protected Key exchange) and SNAPI(Secure Network Authentication with Password Identification) are designed in [7][8]. Very recently, Katz, Ostrovsky and Yung have shown a protocol in the standard model that does not require the random oracle assumption[9]. However, most of the protocols that have security proof for the asymmetric model depends on the ad hoc design and are not practical yet.

We present a systematic and efficient way to transform password authenticated key exchange protocols of the symmetric model into protocols of the password verifier model without losing the security proof.

2. Model and Definitions

We recommend the reader to be familiar with the model of [6] and [9], in which we prove security of our transform. We review the main concepts of their model in this section. For more details, refer to [6].

PROTOCOL PARTICIPANTS, PASSWORDS. We fix a nonempty set ID of principals each of which is either a client $C \in Client$ or a server $S \in Server$. We let $U^{def} = Client \cup Server$. Each $C \in Client$ has a password pw_C that is produced by the *long-lived key generator* PW and each $S \in Server$ holds a vector $pw_S = \langle pw_S[C] \rangle_{C \in Client}$ that contains an entry for each client. In a protocol for the symmetric model, $pw_C = pw_S[C]$. In a protocol for the asymmetric model, $pw_S[C]$ will typically be chosen so that it is hard to compute pw_C from C, S and $pw_S[C]$. The password pw_C(and therefore the $pw_S[C]$) has low entropy.

INITIALIZATION. Before the protocol executes, an initialization process occurs to set passwords for each client and public parameters. Passwords are determined by running an LL-key generator, PW. We assume that password is chosen independently and uniformly at random from $\{1,...,N\}$, where N is a constant, independent of the security parameter.

EXECUTION OF THE PROTOCOL. A protocol determines how instances of the principals behave in response to signals from their environment. It is the adversary who sends these signals. The adversary is assumed to have complete control over all communication in the network. Thus, the adversary's interaction with the principals is modeled via access to oracles whose inputs may range over $U \in User$ and $i \in N$. We call instance i of principal U an oracle and we denote it Π^i_U. The adversary can make queries to any instance which is provided endlessly. The query types that are defined in [6] are:

1. Send(U,I,M) This sends message M to instance Π^i_U. The response that is prescribed by the protocol P is given back and it also outputs the internal states.
2. Execute(C,i,S,j) This query executes the protocol between Π^i_C and Π^j_S and outputs a transcript of this execution. It is useful to model dictionary attacks.
3. Corrupt(U,pw) The adversary obtains pw_U and all states of U.
4. Reveal(U,i) This query outputs the session key sk^i_U, which is in SK, the session key space.
5. Test(U,i) This query is allowed only once. A random bit b is generated; if $b=1$ the adversary is given sk^i_U, and if $b=0$ the adversary obtains a random session key.
6. Oracle(M) The adversary is given the oracle access to the choice of the standard model, the random oracle model, or the ideal cipher model.

AKE ADVANTAGE. In a protocol execution of P,PW,SK,A, the adversary wins if she asks a single Test query, outputs a bit b' and $b'=b$. The ake advantage of A in attacking (P,PW,SK) is defined as $\mathrm{Adv}^{ake}_{P,A} = 2Pr[Win]-1$. A poly-time adversary will be able to break any protocol by attempting to impersonate a user and trying all passwords one-by-one. However, the adversary has

the advantage in attacking the protocol bounded by $O(q_{send}/N)+\varepsilon(k)$ for some negligible function $\varepsilon()$. Thus, if a protocol is said to be secure, it has the proof that the advantage in attacking has the above bound.

DIFFIE-HELLMAN ASSUMPTION. We'll provide security proof under the computational Diffie-Hellman assumption. Let $G = <g>$ be a finite group, also let A be an adversary that outputs a list of group elements. Then we define

$$\mathrm{Adv}_G^{dh}(A) = \Pr[x, y \leftarrow \{1,\ldots,|G|\} : g^{xy} \in A(g^x, g^y)]$$

$$\mathrm{Adv}_G^{dh}(t,q) = \max_A\{\mathrm{Adv}_G^{dh}(A)\},$$

where the maximum is over all adversaries that run in time at most t and output a list of q group elements.

GUILLOUS-QUISQUATER ADVANTAGE. Security of the transformation with Guillous- Quisquater's protocol will be proved under the assumption that simulator of the transcript $T_{P,V}$ of their protocol runs with the following advantage:

$$\mathrm{Adv}_{P,V}^{gq}(A) = 2|E_A(p_0) - E_A(p_1)| - 1,$$

where p_0 is a probability distribution of random transcripts generated by A and p_1 is a probability distribution of genuine transcripts of Guillous-Quisquater's protocol. Here, adversary $A:(X_2)^l \rightarrow \{0,1\}$ tries to decide if a transcript $(z_1,z_2,\ldots,z_l)$ of length l is more likely to have risen from probability distribution p_0 or from probability distribution p_1. For j = 0,1 define

$$E_A(p_j) = \sum_{(z_1,\ldots,z_l)\in(Z_2)^l} p_j(z_1,\ldots,z_l)\,p(A(z_1,\ldots,z_l)=1|(z_1,\ldots,z_l))$$

$$\mathrm{Adv}_{P,V}^{gq}(t) = \max_A\{\mathrm{Adv}_{P,V}^{gq}(A)\},$$

where P,V are honest participants(Prover and Verifier).

3. The Transformation

A zero-knowledge interactive proof-based identification scheme looks like an authentication protocol of the asymmetric model. However, the fact that the password is neither random nor long prevents us from adopting directly ZKIP to password based protocols. That is, an attacker can mount the off-line guessing attack by seeing a transcript even though the public key is kept secret between the prover and the verifier.

Instead of using directly the ZKIP-based identification scheme, we use it to transform a password based authentication protocol of the symmetric model into the protocol of the asymmetric model.

Now, let's denote *s2u-ake(x)* by an. authenticated key exchange protocol for the symmetric model using a shared secret x, where Server does not authenticate User, but User authenticates Server. Using the primitive of *the symmetric model* and a ZKIP-based identification scheme, we show how to construct a *mutual* authentication and key exchange protocol of *the asymmetric model*[1].

Following is the outline of the transform:

1. $C \in Client$ sends a commitment number to Server.
2. Using the password verifier $V = T(pw)$, Client authenticates $S \in Server$ with *s2u-ake*(V) and shares a temporary key *tsk* with Server, where $T()$ is a proper oneway function and depends on the underlying ZKIP-based identification scheme.
3. Server sends a random number r to Client.
4. Client computes the question number by $c = H(r \| tsk)$ and sends a witness number that is computed from the commitment number and the question number. $H()$ is a collision resistant hash function.
5. Server checks the witness number using the commitment number, V and her own c to authenticate Client.

In the above outline, notice that a password verifier V instead of a password *pw* is used in the protocol *s2u-ake(x)*. It means that the protocol to be transformed is used only for the proof of server's knowledge of the password verifier and for temporal key exchange.

As an instance of *s2u-ake(x)*, AddSCA(EKE2) in

1) The transformation does not require mutual authentication of the primitive, but the transformed protocol has the property of mutual authentication. Also, the transformation does not change the security assumption that s2u-ake(x) relies on if it is the ideal cipher model or the random oracle assumption. However, the transform degrades the standard model like [9] into the random oracle model, though it does not mean the degrade of security in the practical aspect.

[6] will be useful. AddSCA (EKE2) is a server to client authentication protocol version of EKE. They show that AddSCA(EKE2) is secure against the dictionary attack in the ideal cipher model[6].

Let's briefly describe the AddSCA(EKE2). Arithmetic is in a finite cyclic group $G = \langle g \rangle$. This group could be $G = Z^*_p$, a prime-order subgroup of Z^*_q, or an elliptic curve group.

1. $C \in Client$ sends $X^* = E_{pw}(g^x)$, where $x \in_R G$ and E_{pw} is symmetric encryption of x with the password pw.
2. $S \in Server$ randomly selects $y \in_R G$ and computes the followings and sends $(auth = H(K' \in 1), Y^*)$ to Client.

$$X = D_{pw}(X^*), K \equiv X^y, Y^* = E_{pw}(g^y), K' = H(K \| X \| g^y \| C \| S)$$

3. Client computes $Y \in D_{pw}(Y^*)$, $K \equiv Y^x$, $K' = H(K \| g^x \| Y \| C \| S)$ and checks $H(K' \| 1) = auth$. If succeeds, Client will be sure that Server knows p^w.
4. If the protocol ends successfully, both Client and Server share a temporary session key $tsk = H(K' \| 0)$.

To be fitted into the outline, the underlying ZKIP-based identification scheme can be any client-initiated protocols with commitment, question and witness revealing stage. For example, Feige-Fiat-Shamir, Guillous-Quisquater, Schnorr's protocols are well fitted[10-12].

4. The Protocol

In this section, we show how to construct the asymmetric version of AddSCA (EKE2) with additional capability of client-to-server authentication using identification schemes such as [10][11][12].

Figure 1 shows the definition of the protocol transformed from AddSCA (EKE2). The description omits some definitions, but they are determined by the underlying ZKIP-based identification scheme. For example, G^* can be the ring in RSA or the multiplicative group in Schnorr or the additive group such as elliptic curves. Also, a (T,F,L) triple is dependent upon the underlying identification scheme.

The formal specification of the protocol including the initialization procedure and queries are described in the appendix A. Security of our protocol is proved assuming the queries defined in the appendix

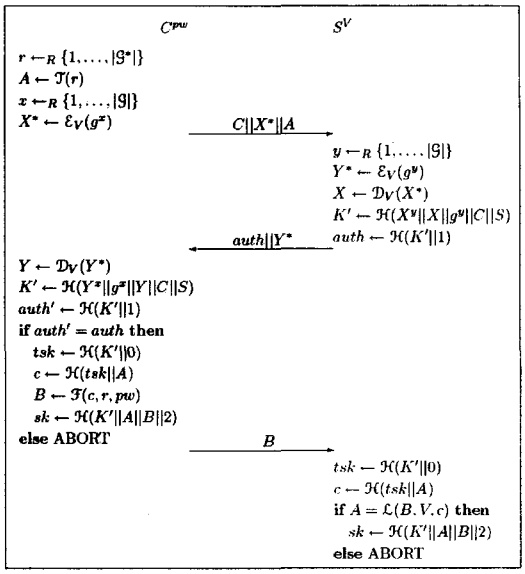

Figure 1 The asymmetric version of AddSCA(EKE2)

Notice that A, the commitment is performed in the first flow. Thus, it is not possible for $C \in Client$ to predict the question c when she decides her test number A. This unpredictability is guaranteed by K that is generated not only by herself but with the collaboration of Server. It is well-known that the prediction of question does break identification systems.

There is a trade-off between performance and security because of $|c|$. If $|c|$ is large, the security level(or, the probability of cheating in the ZKIP literature) is high but more computations are needed. However, the lower bound for $|c|$ is $|PW|$.

Followings are the selected examples.

SCHNORR. If the underlying identification scheme is Schnorr's protocol, $G^* = F_p = \langle \alpha \rangle$, $T(r)^{def} = \alpha^r$ and $L(B,V,c)^{def} = \alpha^B V^c \bmod p$, $F(c,r,pw)^{def} = r + pw * c \bmod q$, where p,q are Schnorr's parameters. The password verifier is computed by $V \equiv \alpha^{-pw} \bmod p$.

FEIGE-FIAT-SHAMIR. Their protocol uses the square root finding problem and has the simulation feature. For the transformation, $G^* = Z^*_n$, $T(r)^{def} = r^2$

mod n and $L(B,V,c)^{def} = B^2 \prod_{i=l,k} v_i^{c_i} \bmod n$, $F(c,r,pw)^{def} = r\prod_{i=1,k}(pw+i)^{c_i} \bmod n$, where n is the RSA modulus. c is the i-th most significant bit of c. The password verifier is computed by $[v_1 \equiv (pw+1)^{-2} \bmod n,\ v^2 \equiv (pw+2)^{-2} \bmod n,\ldots, v_k \equiv (pw+k)^{-2} \bmod n,\ V = H(v_1,v_2,\ldots,v_k)]$.

GUILLOUS-QUISQUATER. Guillous-Quisquater's protocol is a deep coin version of Feige-Fiat-Shamirt's. We define that $G^* = Z^*_n$, $T(r)^{def} = r^e \bmod n$ and $L(B,V,c)^{def} = B^e V^c \bmod n$, $F(c,r,pw)^{def} = r*pw^c \bmod n$, where (n,e) are the Guillous-Quisquater's parameters. The password verifier is computed by $V \equiv pw^{-e} \bmod n$.

Using ZKIP-based identification schemes other than those, one can easily transform AddSCA (EKE2) with the transformation defined in figure 1.

5. Security

We treat security aspects of our protocol in this section. Since AddSCA(EKE2) is secure in the ideal cipher model, the security model that the transformed protocol relies on is also the ideal cipher model. The adversary A can make queries to any instance: she has an endless supply of $\prod_U^i$ oracles($i \in N$, $N(1 \le N \le \sqrt{|G|}/q)$) is the size of the set of Password). $q_{se}, q_{re}, q_{co}, q_{ex}, q_{or}$ count the number of Send, Reveal, Corrupt, Execute, and Oracle queries, respectively. Also, t means the adversary's running time.

We prove the security of the asymmetric version of AddSCA(EKE2) whose underlying identification protocol is Guillous-Quisquater's protocol, which we denote as GQ^*.

Proposition 1. An attacker cannot succeed in the off-line dictionary attack against GQ^* in the ideal cipher model with the advantage greater than

$$\mathrm{Adv}^{ake-fs}_{GQ^*,PW,SK}(t,q_{se},q_{re},q_{co},q_{ex},q_{or})$$
$$\le \frac{2q_{se}}{N} + q_{se}\cdot q_{or}\cdot \mathrm{Adv}^{dh}_{G}(t',q_{or}) + q_{se}\cdot q_{or}\cdot \mathrm{Adv}^{eq}_{P,V}(t',q_{or}) + \frac{O(q_{to}^2)}{|G|} + \frac{O(1)}{\sqrt{|G|}},$$

where $t' = t + O(q_{se} + q_{or})$ and $q_{to} = q_{se} + q_{re} + q_{co} + q_{ex} + q_{or}$

Proof To evaluate adversary's advantage against our protocol, we divide adversary's advantages into the following 3 cases:

1. Invalid messages sent to an instance $\prod_U^i$ does not help an adversary to increase the advantage of success.
2. Advantage for an adversary to make valid and new messages is negligible.
3. Even though valid but previously used messages are used(replay attack), added advantage is negligible.

Here, a *valid* message is not defined in itself. For the first message and the corresponding third message pair to be valid, A and B must satisfy $A \equiv L(B,V,c)$ relationship for a given c. Also for the second message to be valid, its *auth* field must be computed from its Y and $D_V(X^*)$ in the corresponding first message.

After the adversary completes sufficient queries according to the above cases, her advantage is evaluated using Test query by considering how much the probability that she guesses correctly b is.

First, consider the case where invalid messages are sent to an instance $\prod_U^i$, where $U \in Server$.

Lemma 1. If an invalid message is sent to an instance $\prod_U^i$, $U \in Server$ by an adversary, the resulting session keys between $\prod_U^i$ and the adversary match with probability of $1/|G|$.

Proof Under the ideal cipher model, distribution of $D_{V'}(E_{V'}(g^x) = g^{x'}$ is uniformly random over G. Similarly, $D_{V'}(E_V(g^y) = g^{y'}$ is also uniformly distributed over G. Adversary's session key is computed from $g^{x'y}$, whereas $\prod_U^i$, $U \in Server$'s session key is from $g^{xy'}$. Thus, probability of $g^{xy'} \equiv g^{xy'}$ is $1/|G|$. □

Similarly, probability that an invalid message to an instance $\prod_U^i$, $U \in Client$ causes same session key generation between an adversary and $\prod_U^i$, $U \in Client$ is $1/|G|$.

Second, let us examine the advantage of an adversary when she makes a new and valid message. When an adversary tries to make a new and valid second message, she has an advantage of $Adv_G^{dh}(t + O(q_{se} + q_{or}), q_{or})$, owing to [6]. Because the number of queries for which the decrease of size in the set of remaining passwords occurs is bounded by q_{se} for q_{or} queries, the advantage of the adversary

increases by $q_{se}q_{or}\ Adv_{C}^{dh}(t + O(q_{se} + q_{or}),q_{or})$. Whenever she tries to make a new and valid first message, she cannot compute the valid third message owing to the simulatability property of Guillouis-Quisquater's protocol. The advantage of success is increased only by $Adv_{P,V}^{gq}(t/,q_{or})$. The protocol includes another messages but Guillouis-Quisquater's protocol's messages, so more exact simulator for the transcript is needed for calculation of the advantage. For the proof of the advantage, we define a transformation GQ^{+} to the original protocol GQ^{*}, and bound the effect the transformation has on the adversary's advantage. This gives us an explicit bound on the adversary's advantage in the original protocol. In protocol GQ^{+}, calls to the Execute oracle are answered as before, but A and B are chosen at random from Z_n^{*}. We call the modified oracle Execute+.

Execute(C,i,S,j) –
$x,y \leftarrow \{1,\ldots,|S|\}\ \ r \leftarrow \{1,\ldots,|S^{*}|\}$
$X \leftarrow g^{x}, X^{*} \leftarrow \mathcal{E}_V(g^{x})$
$A \leftarrow \mathcal{T}(r), msg-out_1 \leftarrow C||X^{*}||A)$
$Y \leftarrow g^{y}, Y^{*} \leftarrow \mathcal{E}_V(g^{y}), K' \leftarrow \mathcal{H}(K||X||Y||C||S)$
$auth \leftarrow \mathcal{H}(K'||1), msg-out_2 \leftarrow auth||Y^{*}$
$tsk \leftarrow \mathcal{H}(K'||0), c \leftarrow \mathcal{H}(tsk||A), B \leftarrow \{1,\ldots,|\mathbf{Z}_n^{*}|\}$
$msg-out_3 \leftarrow B$
$sk_C^{i} \leftarrow sk_S^{j} \leftarrow \mathcal{H}(K'||A||B||2), sid_C^{i} \leftarrow sid_S^{j} \leftarrow C||X^{*}||S||Y^{*}$
return $(msg-out_1, msg-out_2, msg-out_3)$

Figure 2 Specification of the Execute+: *s2c-ake()* = AddSCA(EKE2)

Lemma 2. The adversary's success probability in GQ^{+} differs by at most $q_{se}\cdot q_{or}\cdot\ Adv_{P,V}^{gq}(t + O(q_{se} + q_{or}),q_{or})$ from its advantage in GQ^{*}.

Proof In the adversary's point of view, Execute and Execute+ behaves similarly. How much she can distinguish transcripts of Execute oracle from ones of Execute+ oracle decides the advantage. Note that because the underlying AddSCA(EKE2) gives an adversary only negligible advantage[6], an adversary cannot find what the question c is.

Consider the transcript

$$T_{GQ^{*}} = ((X, A' \equiv r_1^{e} \in_R Z_n^{*});(Y, auth);(B' \in_R Z_n^{*}))$$

which is a transcript from the oracle Execute+.

Now, compare $T_{GQ^{+}}$ with the following transcript:

$$T_{GQ^{*}} = ((X, A \equiv r^{e} \bmod n);(Y, auth);(B \in r * pw^{c} \bmod n))$$

which is a transcript from the oracle Execute.

Instead of observing B and B', we may compare $V^{c} \in A/B^{e} \bmod n$ and $A'/B'^{e} \equiv r_1^{e} \equiv r_2^{c} \equiv R^{c} \bmod n$ for some, $r_1,r_2,\mathrm{R} \in_{\mathrm{R}} Z_n^{*}$ where $V \equiv pw^{-2} \bmod n$. Consider random variables $L_1(i) = V^{\pi(i)} \bmod n$ and $L_2(i) = R^{\pi(i)} \bmod n$, where $\pi(\mathrm{i})$ is a permutation function, $\pi: x \rightarrow y; x,y \in Z_e^{*}$. Here, remind that c is not known to the adversary A by the assumption, so $\pi(\mathrm{i})$ is not known to her. We can state that because $\pi(\mathrm{i})$ is a random permutation and it is hidden from A, A cannot distinguish $L_1(i)$ from $L_2(i)$. Thus, this problem of distinguishing $T_{GQ^{*}}$ from $T_{GQ^{+}}$ falls into the Guillous-Quisquater's simulation problem and the random variable $T_{GQ^{+}}$ is computationally distinguishable from $T_{GQ^{*}}$ with advantage greater than $Adv_{P,V}^{gq}(t,q)$. Because the number of queries for which the decrease of size in the set of remaining passwords occurs is bounded by q_{se} for q_{or} queries, the advantage of the adversary increases by $q_{se}\cdot q_{or}\cdot\ Adv_{P,V}^{gq}(t' + q_{or})$. □

The following corollary bounds the effect on the adversary's advantage.

Corollary 1 An adversary's probability of sending, at any point during the protocol, a first and its corresponding third message which is both new and valid is bounded by $q_{se}q_{or}\ Adv_{P,V}^{gq}(t + O(q_{se} + q_{or},q_{or})$ □

Finally, we consider the advantage when an adversary performs replay attack.

Lemma 3. If valid but previously used messages are used (replay attack) by an adversary, added advantage is bounded by

$$\frac{q_{se}}{N} + \frac{1}{|G|}$$

Proof Soundness property of Guillous-Quisquater's protocol directly explains the first term. That is, if an adversary A performs Send(U,i,M_{used}) oracle queries to $\prod_{U}^{i}$, $U \in Server$ using previously used messages M_{used}, the probability that the corresponding question c in current session matches the question embedded in the previously used messages is $1/N$ per one Send(U,i,M_{used}) query. If the questions match, A can pretend to be a legal user.

Secondly, if an adversary A performs Send(U,i,M_{used}) oracle queries to $\prod^{i}_{U}$, $U \in Client$ using previously used messages ($auth \| Y^*$), the probability that auth is valid depends on X in the first message. A has at most $1/|G|$ probability to succeed. Thus, the added advantage is bounded by $(q_{se}/N) + (1/|G|)$. □

The adversary's advantage in the protocol is, thus, bounded by the expression of proposition 1. □

Using the lemma 2, we can connect our security result with Bellare-Pointcheval-Rogaway's security result[6] as following:

Proposition 2. If there is an algorithm A^* that can perform off-line dictionary attack with the advantage greater than $Adv^{ake-fs}_{GQ^*,PW,SK}$ against GQ^* in the ideal cipher model, there is an algorithm A^+ that can solve AddSCA(EKE2) with the advantage greater than $Adv^{ake-fs}_{GQ^*,PW,SK}$.

Proof Let's assume the algorithm A^* that can perform off-line dictionary attack with the advantage greater than $Adv^{ake-fs}_{GQ^*,PW,SK}$ against GQ^* in the ideal cipher model. Also, we know that AddSCA(EKE2) is secure[6].

We show that if such an algorithm A^* exists, there exists an algorithm A^+ that can solve AddSCA(EKE2) with the advantage greater than $Adv^{ake-fs}_{GQ^*,PW,SK}$.

A transcript of GQ^* consists of three flows: (X,$A \equiv r^e \bmod n$) (Y, $auth$) ($B \equiv r*f(S)^c \bmod n$)), which will be fed into A^*. Then, A^* will find out what the secret is with the advantage greater than $Adv^{ake-fs}_{GQ^*,PW,SK}$ according to the assumption.

Let $T_{eke} = ((X);(Y,auth))$ be a transcript AddSCA (EKE2) of which secret A^+ would like to find out. A^+ works in the following way:

1. A^+ makes a transcript $T_{GQ^+} = ((X,A' \equiv r_1{}^e \in {}_RZ_n^*;(Y,auth);(B' \in {}_RZ_n^*))$
2. It feeds the transcript T_{GQ^+} into A^*. According to lemma 2, A^* cannot distinguish T_{GQ^+} from T_{GQ^*} with advantage more than $Adv^{ake-fs}_{GQ^*,PW,SK}$. Because of the assumption, A^* will ouput the secret with the advantage greater than $Adv^{ake-fs}_{GQ^*,PW,SK}$. A^+ takes the output, and returns it.

It takes $O(1)$ time to make the transcript T_{GQ^+} from a transcript T_{eke}, so there exists the algorithm A^+ that can find out a secret S from T_{eke} with the advantage greater than $Adv^{ake-fs}_{GQ^*,PW,SK}$. This contradicts the proof by [6] that AddSCA(EKE2) is strong against off-line guessing attack. Thus, if AddSCA(EKE2) is strong against off-line guessing attack, GQ^* is secure against off-line guessing attack in the ideal cipher model and under the random oracle assumption. Loosely speaking though pw is in itself weak, the entropy of B is amplified by c, because c as well as pw is hidden from A^*.

□

Proposition 3. Honest *Server* cannot get any additional information on pw after the protocol GQ^* completes except the fact that *Client* knows the pw corresponding to $V = T(pw)$ that Server knows with advantage greater than $q_{se} \cdot q_{or} \cdot Adv^{gq}_{P,V}(t' + q_{or})$.

Proof It is clear from the simulation property of the underlying Guillous-Quisquater's protocol.

There exists an oracle that can produce a transcript T_o which is computationally indistinguishable from a transcript T_{GQ^*} without interacting an honest *Client*. Because the oracle has an access to V and the question number c, she can easily construct T_o.

1. The oracle generates random numbers $x,y \in_R G, r \in_R Z_n^*$.
2. Using the random numbers and V, she constructs T_o.

$$T_o = ((E_V(g^x), r^e * V^c \bmod n);(E_V(g^y), auth);(r))$$

T_o is computationally indistinguishable from T_{GQ^*} with advantage greater than $Adv^{gq}_{P,V}(t', q_{or})$, which means that there exists an oracle that produces a transcript with the same probability distribution of T_{GQ^*} without interacting with an honest *Client*. Thus, T_{GQ^*} does not leak any information on the secret except the fact that *Client* knows the secret.

□

The proposition 1, 2 and 3 hold to the other protocols in section4 in a similar manner.

6. Practical Aspect and Conclusion

Using ZKIP-based identification schemes, we presented a generic transformation of protocols for the symmetric model into ones for the asymmetric

model.

Our proposalis very practical compared to other protocols, since only the amount of computation for an underlying ZKIP-based identification scheme, the amount of computation for one Diffie-Hellman key exchange, nine times of computation of a hash function and two symmetric encryptions/decryptions are needed. Generally, the amount of computation for a ZKIP-based identification scheme is much less than that for a public key encryption/ decryption or additional Diffie-Hellman key exchange is. Also, if we choose a proper finite cyclic group G such as an elliptic curve group, the added cost will be substantially decreased.

For the GQ case, our proposal requires 2 exponentiations of client and 2.75 exponentiations of server in parallel, where pre-computation is assumed. In case of Schnorr, it requires only one multiplications of client and 2.75 exponentiations of server. Thus, compared with PAK-X(Password Authenticated Key exchange-X) and SNAPI-X, the amount of computation is quite small. SNAPI-X requires 5 exponentiations of client and 4 exponentiations of server and PAK-X requires 4 exponentiations of both server and client.

Formal description and the security proof only for the protocol using AddSCA(EKE2) are given, but we are working on other protocols such as PPK of [7], SNAPI of [8], and the password-AKE of [9].

References

[1] S. Bellovin and M. Merrit, "Encrypted key exchange: password based protocols secure against dictionary attacks," IEEE Comp. Society Symp. on Research in Security and Privacy, 1992, pp. 7284.

[2] D. Jablon, "Strong password-only authenticated key exchange," ACM Comp. Comm. Review, 1996, Vol. 26, No. 5, pp. 526.

[3] S. Bellovin and M. Merrit, "Augmented encrypted key exchange: a password based protocol secure against dictionary attacks and password file compromise," ACM Conference on Comp. and Comm. Security, 1993, pp. 244-250.

[4] D. Jablon, "Extended Password Key Exchange Protocols Immune to Dictionary Attacks," Proc. of WET-ICE '97, IEEE Computer Society, June, 1997, Cambridge, MA, pp. 248-255.

[5] T. Wu, "Secure Remote Password Protocol," Internet Society Symp. Network ad Distributed System Security, 1998.

[6] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," Proceedings of EuroCrypt 2000, Lecture Notes in Computer Science, Springer-Verlag, 2000, pp. 139-155.

[7] V. Boyko, P. MacKenzie and S. Patel, "Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman," Proceedings of EuroCrypt 2000,Lecture Notes in Computer Science, Springer-Verlag, 2000, pp. 156-171.

[8] P. MacKenzie, S. Patel and R. Swaminathan, "Password-Authenticated Key Exchange Based on RSA," Proceedings of Asiacrypt 2000, Lecture Notes in Computer Science, Springer-Verlag, 2000, pp. 599-613.

[9] J. Katz, R. Ostrovsky and M. Yung, "Efficient Password-Authenticated Key Exchange Uinsg Human-Memorable Passwords,"Proceedings of Eurocrypt 2001, Lecture Notes in Computer Science, Springer-Verlag, 2001, pp. 475-494.

[10] U. Feige, A. Fiat and A. Shamir, "Zero-knowledge proofs of identity," Journal of Cryptology, Vol. 1, No. 2, 1988, pp. 77-94.

[11] L.C. Guillou and J.J. Quisquater, "Protocol fitted to security microprocessor minimizing both transmission and memory," Proceedings of EuroCrypt 88,Lecture Notes in Computer Science, Springer-Verlag, 1988, pp. 123-128.

[12] C.P. Schnorr, Efficient Identification and Signatures for Smart cards, Advances in Cryptology : Proceedings of Crypto 89, Lecture Notes in Computer Science, Springer-Verlag, New York, 1989, pp. 239-251.

A. Formal Specification of the Protocol

Initialization(1^k) –
$\mathcal{H} \xleftarrow{R} \Omega$; $\langle pw_C, V_S = \mathcal{F}(pw_C)\rangle_{C\in Client, S\in Server} \xleftarrow{R} PW$
Choose $\mathcal{G}$ of which size is k and a random generator $g \leftarrow \mathcal{G}$
Publish parameters $(\mathcal{G}, \mathcal{G}^*, \mathcal{H}, \mathcal{T}, \mathcal{F}, \mathcal{L}, \mathcal{E}. \mathcal{D}, g)$

Execute(C, i, S, j) –
$x, y \leftarrow \{1, \ldots, |\mathcal{G}|\}$ $r \leftarrow \{1, \ldots, |\mathcal{G}^*|\}$
$X \leftarrow g^x, X^* \leftarrow \mathcal{E}_V(g^x)$
$A \leftarrow \mathcal{T}(r), msg - out_1 (\leftarrow C||X^*||A)$
$Y \leftarrow g^y, Y^* \leftarrow \mathcal{E}_V(g^y), K' \leftarrow \mathcal{H}(K||X||Y||C||S)$
$auth \leftarrow \mathcal{H}(K'||1), msg - out_2 (\leftarrow auth||Y^*$
$tsk \leftarrow \mathcal{H}(K'||0), c \leftarrow \mathcal{H}(tsk||A), B \leftarrow \mathcal{F}(c, r, pw)$
$msg - out_3 \leftarrow B$
$sk_C^i \leftarrow sk_S^j \leftarrow \mathcal{H}(K'||A||B||2), sid_C^i \leftarrow sid_S^j \leftarrow C||X^*||S||Y^*$
return $(msg - out_1, msg - out_2, msg - out_3)$

Reveal(U, i) –
return sk_U^i

Test(U, i) –
$b \xleftarrow{R} \{0, 1\}, sk \leftarrow SK$
if $b = 0$ **then return** sk **else return** sk_U^i

Oracle(M) –
return $h(M)$

Corrupt(U) –
return $(pw_U, \{state_U^i\}_{i\in N})$

Send(U, i, M) –
if $state = READY$ and $U \in Client$ **then**
  $\langle C \rangle \leftarrow U$ $\langle S \rangle \leftarrow msg - in$, where $S \in Server$
  $x \leftarrow \{1, \ldots, |\mathcal{G}|\}$ $X \leftarrow g^x$ $X^* \leftarrow \mathcal{E}_V(g^x)$
  $r \leftarrow \{1, \ldots, |\mathcal{G}^*|\}$ $A \leftarrow \mathcal{F}(r)$ $msg - out \leftarrow C||X^*||A$
  $tsk \leftarrow sid \leftarrow pid \leftarrow sk \leftarrow \varepsilon$ $acc \leftarrow term \leftarrow FALSE$ $state \leftarrow \langle X, S \rangle$
  return$(msg - out, acc, term, sid, pid, tsk, sk, state)$
else if $state = READY$ and $U \in Server$ **then**
  $\langle S \rangle \leftarrow U$ $\langle C||X^*||A \rangle \leftarrow msg - in$, where $C \in Client$ X^* is
  a ciphertext and A is a commitment.
  $y \leftarrow \{1, \ldots, |\mathcal{G}|\}$ $Y \leftarrow g^y, Y^* \leftarrow \mathcal{E}_V(g^y)$ $X \leftarrow \mathcal{D}_V(X^*)$
  $K \leftarrow X^y$ $K' \leftarrow \mathcal{H}(K||X||Y||C||S)$ $auth \leftarrow \mathcal{H}(K'||1)$
  $msg - out \leftarrow Y^*||auth$ $sid \leftarrow C||X^*||S||Y^*$ $pid \leftarrow C$ $tsk \leftarrow \mathcal{H}(K'||0)$
  $sk \leftarrow \varepsilon$ $acc \leftarrow term \leftarrow FALSE$ $state \leftarrow \langle y, C \rangle$
  return$(msg - out, acc, term, sid, pid, tsk, sk, state)$
else if $state = \langle x, S \rangle$ and $U \in Client$ **then**
  $\langle C \rangle \leftarrow U$ $\langle Y^*, auth \rangle \leftarrow msg - in$, where Y^* is a ciphertext.
  $Y \leftarrow \mathcal{D}_V(Y^*)$ $K \leftarrow Y^x$ $K' \leftarrow \mathcal{H}(K||X||Y||C||S)$ $auth' \leftarrow \mathcal{H}(K'||1)$
  $sid \leftarrow C||X^*||S||Y^*$ $pid \leftarrow S$ $tsk \leftarrow \varepsilon$
  if $auth = auth'$ **then**
    $tsk \leftarrow \mathcal{H}(K'||0)$ $c \leftarrow \mathcal{H}(tsk||A)$
    $B \leftarrow \mathcal{F}(c, r, pw)$ $sk =\leftarrow \mathcal{H}(K'||A||B||2)$
    $msg - out \leftarrow B$ $acc \leftarrow term \leftarrow TRUE$ $state \leftarrow DONE$
  else
    $sk \leftarrow \varepsilon$ $acc \leftarrow term \leftarrow FALSE$ $state \leftarrow DONE$
  return$(msg - out, acc, term, sid, pid, tsk, sk, state)$
else if $state = \langle y, C \rangle$ and $U \in Server$ **then**
  $\langle S \rangle \leftarrow U$ $\langle B \rangle \leftarrow msg - in$, where B is a witness number.
  $c \leftarrow \mathcal{H}(tsk||A)$ $sid \leftarrow C||X^*||S||Y^*$ $pid \leftarrow C$
  if $A = \mathcal{L}(B, V, c)$ **then**
    $sk \leftarrow \mathcal{H}(K'||A||B||2)$ $acc \leftarrow term \leftarrow TRUE$ $state \leftarrow DONE$
  else
    $sk \leftarrow \varepsilon$ $acc \leftarrow FALSE$ $term \leftarrow TRUE$ $state \leftarrow DONE$
  return$(msg - out, acc, term, sid, pid, tsk, sk, state)$

Figure 3 Specification of the Protocol: *s2c-ake()* = AddSCA(EKE2)

양 대 헌

1994년 2월 KAIST 전기 및 전자 공학과 학사. 1996년 2월 연세대학교 컴퓨터과학과 석사. 2000년 8월 연세대학교 컴퓨터 과학과 박사. 2000년 9월~2003년 2월 ETRI 정보보호연구본부 선임연구원 2003년 3월~현재 인하대학교 정보통신대학원 전임강사. 관심분야는 암호이론, 암호프로토콜, 인증 프로토콜, 무선 인터넷 보안