社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006 — CSEC — 34 (46)
2006／7／21

An Efficient Anonymous Password-Authenticated Key Exchange Protocol

辛 星漢[†] 古原 和邦[†] 今井 秀樹[†,††]

[†] 〒101-0021 東京都千代田区外神田 1-18-13 産業技術総合研究所情報セキュリティ研究センター
[††] 〒112-8551 東京都文京区春日 1-13-27 中央大学理工学部 電気電子通信工学科

E-mail: [†]seonghan.shin@aist.go.jp

あらまし Recently, Viet et al., [21] have proposed an anonymous password-authenticated key exchange (PAKE) protocol against a passive server, who follows the protocol honestly but it is curious about identity of client. In this paper, we propose an efficient construction for anonymous PAKE protocol (we call it the EAP protocol) which provides semantic security of session keys in the random oracle model, with the reduction to the computational Diffie-Hellman problem, as well as anonymity against a passive server. Specially, the EAP protocol has about 50% reduction (compared to [21]) in the number of modular exponentiations for both client and server, and its communication bandwidth for the modular size of prime p is independent from the number of clients while [21] is not.

キーワード 鍵交換プロトコル、パスワード、オンライン攻撃、オフライン攻撃、匿名

An Efficient Anonymous Password-Authenticated Key Exchange Protocol

SeongHan SHIN[†], Kazukuni KOBARA[†], and Hideki IMAI[†,††]

[†] Research Center for Information Security, AIST, 1-18-13 Sotokannda, Chiyoda-ku, Tokyo 101-0021 Japan
[††] Chuo University

E-mail: [†]seonghan.shin@aist.go.jp

Abstract Recently, Viet et al., [21] have proposed an anonymous password-authenticated key exchange (PAKE) protocol against a passive server, who follows the protocol honestly but it is curious about identity of client. In this paper, we propose an efficient construction for anonymous PAKE protocol (we call it the EAP protocol) which provides semantic security of session keys in the random oracle model, with the reduction to the computational Diffie-Hellman problem, as well as anonymity against a passive server. Specially, the EAP protocol has about 50% reduction (compared to [21]) in the number of modular exponentiations for both client and server, and its communication bandwidth for the modular size of prime p is independent from the number of clients while [21] is not.

Key words authenticated key exchange, passwords, on-line and off-line dictionary attacks, anonymity

1. Introduction

Since the discovery of the Diffie-Hellman key exchange [9], many researchers have tried to design a cryptographic protocol for realizing secure channels. This kind of protocol is necessary because higher-level protocols are frequently developed and analyzed assuming the existence of secure channels between all parties. In the 2-party setting (e.g., a client and a server), this can be achieved by an authenticated key exchange (AKE) protocol at the end of which the two parties authenticate each other and share a common (and temporal) session key to be used for subsequent cryptographic algorithms (e.g., AES-CBC or MAC). For authentication, the parties typically share some information in advance. The shared information may be the form of high-entropy cryptographic keys: either a secret key that can be used for symmetric-key encryption or message authentication code (e.g., [7], [20]), or public keys (exchanged by the parties, while the corresponding private keys are kept secret) which can be used for public-key encryption or digital signatures (e.g., [2], [10], [16], [20], [23]).

In practice, high-entropy keys may often and commonly be substituted by low-entropy human-memorable passwords chosen from a relatively small size of dictionary (e.g., alphanumerical passwords). Owing to the usability of passwords, password-based AKE protocols have been extensively investigated for a long time where a client remembers a short password and the corresponding server holds the password or its verification data that is used to verify the client's knowledge of the password. However, there exist two major attacks on passwords: on-line and off-line dictionary attacks. The on-line dictionary attack is a series of exhaustive searches for a secret performed on-line, so that an adversary can sieve out possible secret candidates one by one communicating with the target party. In contrast, the off-line dictionary attack is performed off-line massively in parallel where an adversary exhaustively enumerates all possible secret candidates, in an attempt to determine the correct one, by simply guessing a secret and verifying the guessed secret with recorded transcripts of a protocol. While on-line attacks are applicable to all of the password-based protocols equally, they can be prevented by letting a server take appropriate intervals between invalid trials. But, we cannot avoid off-line attacks by such policies, mainly because the attacks can be performed off-line and independently of the server. At first sight, it seems paradoxical and more difficult to design a secure AKE protocol for the password-based setting partly because that has to "bootstrap" from a weak shared secret to a strong one.

1.1 Previous Works: AKE Protocols with Anonymity

Natural AKE protocols one can think of are those (e.g., SSL/TLS [11], [15]) based on PKI (Public Key Infrastructures). In the password-based user authentication mode of SSL/TLS, a client remembers his/her password and holds a server's public key whereas the corresponding server has password verification data and its private key. The AKE protocol works as follows: the parties first establish a secure channel with the server's public key(注1) and then the password is transmitted through the channel. In [16], Krawczyk discussed carefully anonymity of the previous PKI-based AKE and their SIGMA protocols. One can easily see that these kind of protocols guarantee client's anonymity against an outside adversary who fully controls the networks.

Without the use of PKI, Lomas et al., proposed AKE protocols with heuristic discussion of resistance to off-line dictionary attacks where a client remembers his/her password and holds a server's public key in advance whereas the corresponding server has password verification data and its private key [18]. This kind of AKE protocols were further studied by Gong [12], and Halevi and Krawczyk [13] gave formal definitions and rigorous proofs of security in the setting. Very recently, Kolesnikov et al., pointed out a subtle flaw in the Halevi and Krawczyk's protocol and then generalized the model by introducing another shared secret (i.e., MAC key) in addition to password and (public, private) key pair [17]. The interesting point is that this kind of AKE protocols don't suffer from off-line dictionary attacks at

(注1) : Before running the actual protocol, the client must verify the server's certificate via CRL (Certificate Revocation Lists) or OCSP (Online Certificate Status Protocol).

the expense of storing the server's public key beforehand. These AKE protocols can also preserve client's anonymity with the same way as the above paragraph (by encrypting client's identity with the server's public key).

Bellovin and Merritt [4] brought up an interesting problem about how to design a secure password-only protocol where a client remembers his/her password only (without any device and any additional assumption) and the counterpart server has password verification data. Their proposed protocols are good examples (though some are turned out insecure) that a combination of symmetric and asymmetric cryptographic techniques can provide insufficient information for an adversary to verify a guessed password and thus defeats off-line dictionary attacks. Later, their AKE protocols have formed the basis for what we call Password-Authenticated Key Exchange (PAKE) protocols(注 2). However, client's anonymity is another problem in that client should send his/her identity first in order to authenticate each other and share a master-secret (to be used for authenticator and session key). Recently, Viet et al., [21] have proposed an anonymous PAKE protocol that combines a PAKE protocol for generating secure channels with an Oblivious Transfer (OT) protocol for client's anonymity. The anonymity is guaranteed against an outside adversary as well as a passive server, who follows the protocol honestly but it is curious about identity of client involved with the protocol. As an application, one may think of a company's public bulletin board to which any employer can upload opinions in a password-based authenticated and anonymous way. Another example can be found in [21].

1.2 Our Contributions

Our motivation is simple: "Is it possible to design a more efficient anonymous PAKE protocol?" In this paper, we give a positive answer with an efficient construction for anonymous PAKE protocol (we call it the EAP protocol) which modifies the underlying PAKE protocol [1], as a component, used in [21] as well. We also show that the EAP protocol provides semantic security of session keys by proving its security in the random oracle model with the reduction to the computational Diffie-Hellman problem. One might think that what makes the EAP protocol more efficient than the original anonymous PAKE one. The main difference comes from the fact that we do not need to apply the OT protocol [22] as it is. Clearly, Theorem 2 shows that our construction is enough to guarantee client's anonymity against a passive server under the same definition of [21]. In addition, we can state the numerical improvements for efficiency: the EAP protocol has about 50% reduction (compared to [21]) in the number of modular exponentiations for both client and server, and its communication bandwidth for the modular size of prime p is independent from the number of clients while [21] is not (see Table 1).

1.3 Organization

This paper is organized as follows. In Section 2., we propose an efficient anonymous PAKE (EAP) protocol. Section 3. and 4. are devoted to its model and security proofs, followed by some comparisons of efficiency in Section 5.. Finally, we conclude in Section 6..

2. An Efficient Anonymous PAKE (EAP) Protocol

In this section, we propose an efficient anonymous PAKE (for short, EAP) protocol that provides semantic security of session keys as well as anonymity against a passive server who follows the protocol honestly, but it is curious about identity of a client involved with the protocol.

2.1 Preliminary

We give some notation to be used. Let G be a finite, cyclic group of prime order q and g be a generator of G (quadratic residues modulo p where $p = aq + 1$) where the Diffie-Hellman problem is hard. Let h be another generator of G so that its discrete logarithm problem with g (i.e., computing $b = \log_g h$) should be hard. These parameters are given as public information. In the aftermath, all the subsequent arithmetic operations are performed in modulo p unless otherwise stated.

Let l denote the security parameter for hash functions. Let N be a dictionary size of passwords. Let

(注 2) : Such protocols are in standardization of IEEE P1363.2 [14]

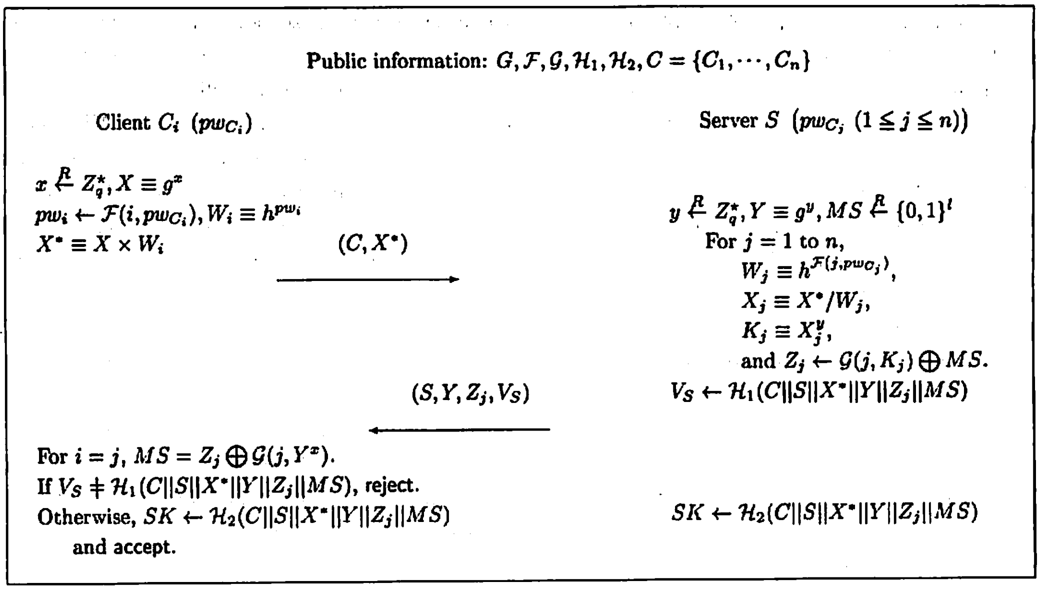
Public information: $G, \mathcal{F}, \mathcal{G}, \mathcal{H}_1, \mathcal{H}_2, C = \{C_1, \cdots, C_n\}$
Client C_i (pw_{C_i})
Server S $(pw_{C_j}$ $(1 \leq j \leq n))$
$x \xleftarrow{R} Z_q^*, X \equiv g^x$
$pw_i \leftarrow \mathcal{F}(i, pw_{C_i}), W_i \equiv h^{pw_i}$
$X^* \equiv X \times W_i$
(C, X^*)
$y \xleftarrow{R} Z_q^*, Y \equiv g^y, MS \xleftarrow{R} \{0,1\}^l$
For $j = 1$ to n,
$W_j \equiv h^{\mathcal{F}(j, pw_{C_j})}$,
$X_j \equiv X^*/W_j$,
$K_j \equiv X_j^y$,
and $Z_j \leftarrow \mathcal{G}(j, K_j) \bigoplus MS$.
$V_S \leftarrow \mathcal{H}_1(C||S||X^*||Y||Z_j||MS)$
(S, Y, Z_j, V_S)
For $i = j$, $MS = Z_j \bigoplus \mathcal{G}(j, Y^x)$.
If $V_S \neq \mathcal{H}_1(C||S||X^*||Y||Z_j||MS)$, reject.
Otherwise, $SK \leftarrow \mathcal{H}_2(C||S||X^*||Y||Z_j||MS)$
and accept.
$SK \leftarrow \mathcal{H}_2(C||S||X^*||Y||Z_j||MS)$

図 1 An efficient anonymous PAKE (for short, EAP) protocol

$\{0,1\}^*$ denote the set of finite binary strings and $\{0,1\}^l$ the set of binary strings of length l. If D is a set, then $d \xleftarrow{R} D$ indicates the process of selecting d at random and uniformly over D. Let "||" denote the concatenation of bit strings in $\{0,1\}^*$. Let "$\bigoplus$" denote the exclusive-OR (XOR) operation of bit strings. The hash function $\mathcal{F}$ is a full-domain hash function, mapping $\{0,1\}^*$ to Z_q^*. The other hash functions are denoted $\mathcal{G}, \mathcal{H}_k : \{0,1\}^* \rightarrow \{0,1\}^l$, for $k = 1, 2$ and 3, where $\mathcal{H}_k$ are distinct secure one-way hash functions (e.g., SHA-256 or RIPEMD-160) one another. Let C and S be the identities of a set of clients and server, respectively, with each ID $\in \{0,1\}^*$.

2.2 The Protocol

Here we assume that each client C_i of the set C has registered a password pw_{C_i} to a server S and the latter maintains a database of clients' passwords pw_{C_j} $(1 \leq j \leq n)$. For simplicity, we assign the clients consecutive integer i $(1 \leq i \leq n)$ where C_i can be regarded as the i-th client.

When client C_i wants to share a session key securely with server S without revealing his identity, he first chooses a random number x from Z_q^* and computes the Diffie-Hellman public value $X \equiv g^x$. The latter is masked in a way of the product of the public value with the verification data W_i where $W_i \equiv h^{pw_i}$ and pw_i is the output of $\mathcal{F}(i, pw_{C_i})$. Then the resultant value is sent to the server together with the identity of a set of clients. On the other hand, server S chooses a random number y from Z_q^* and a random master-secret MS from $\{0,1\}^l$, and computes its Diffie-Hellman public value $Y \equiv g^y$. With the message from client C_i, the server computes X_j, by dividing the received masked public value with each of the verification data, and Z_j that is derived from XORing MS and the hash of each Diffie-Hellman key. Also server S generates an authenticator and a session key, both of which are just the hash of some values and the master-secret MS, and then sends its identity, the Diffie-Hellman public value, $\{Z_j\}_{1 \leq j \leq n}$ and the authenticator. After computing MS from Y and Z_j in an obvious way, client C_i verifies the received authenticator V_S prior to generating his session key. In order to avoid so-called partition attacks [19], [24], both of client and server should check the subgroup order of Y and X^*, respectively: $Y^q \equiv (X^*)^q \equiv 1$. The graphical description of the whole protocol appears in Fig. 1.

3. The Model and Security Notions

In this section we introduce the model based on

[3], [5] and security notions.

3.1 The Model

We denote by C_i and S two parties that participate in the key exchange protocol P. Each of them may have several instances called oracles involved in distinct, possibly concurrent, executions of P where we denote C_i (resp., S) instances by C^i (resp., S^j), or by U in case of any instance. In the EAP protocol, client C_i and server S share a low-entropy secret pw_{C_i}, drawn from a small dictionary of password D_{Password}, whose cardinality is N. During the protocol, an adversary has the entire control of the network. Let us show the capability of adversary $\mathcal{A}$ each query captures:

- $\mathsf{Execute}(C^i, S^j)$: This query models passive attacks, where the adversary gets access to honest executions of P between the instances C^i and S^j by eavesdropping.
- $\mathsf{Send}(U, m)$: This query models active attacks by having $\mathcal{A}$ send a message to instance U. The adversary $\mathcal{A}$ gets back the response U generates in processing the message m according to the protocol P. A query $\mathsf{Send}(C^i, \mathtt{Start})$ initializes the key exchange protocol, and thus the adversary receives the initial flow.
- $\mathsf{Reveal}(U)$: This query handles the misuse of the session key (e.g., use in a weak symmetric-key encryption) by any instance U. The query is only available to $\mathcal{A}$ if the instance actually holds a session key and the latter is released to $\mathcal{A}$.
- $\mathsf{Test}(U)$: This oracle is used to see whether or not the adversary can obtain some information on the session key by giving a hint on the key. The Test-query can be asked at most once by the adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the instance U is "fresh" in that the session key is not obviously known to the adversary. This query is answered as follows: one flips a (private) coin $b \in \{0, 1\}$ and forwards the corresponding session key SK ($\mathsf{Reveal}(U)$ would output) if $b = 1$, or a random value except the session key if $b = 0$.

3.2 Security Notions

The adversary $\mathcal{A}$ is provided with random coin tosses, some oracles and then is allowed to invoke any number of queries as described above, in any order. The aim of the adversary is to break the privacy of the session key (a.k.a., semantic security) or the authentication of the parties in the context of executing P.

The AKE security is defined by the game $\mathbf{Game}^{\mathsf{ake}}(\mathcal{A}, P)$, in which the ultimate goal of the adversary is to guess the bit b involved in the Test-query by outputting this guess b'. We denote the AKE advantage, by $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$, as the probability that $\mathcal{A}$ can correctly guess the value of b. The protocol P is said to be (t, ε)-AKE-secure if $\mathcal{A}$'s advantage is smaller than ε for any adversary $\mathcal{A}$ running time t.

Another goal is to consider unilateral authentication of either C_i (C_i-auth) or S (S-auth) wherein the adversary impersonates a party. We denote by $\mathsf{Succ}_P^{\mathsf{C_i-auth}}(\mathcal{A})$ (resp., $\mathsf{Succ}_P^{\mathsf{S-auth}}(\mathcal{A})$) the probability that $\mathcal{A}$ successfully impersonates an C_i instance (resp., an S instance) in an execution of P, which means that S (resp., C_i) agrees on a key while the latter is shared with no instance of C_i (resp., S). A protocol P is said to be (t, ε)-Auth-secure if $\mathcal{A}$'s success probability for breaking either C_i-auth or S-auth is smaller than ε for any adversary $\mathcal{A}$ running time t.

By following the definition of anonymity from [21], we can say that a protocol P is *anonymous* if a passive server cannot get any information about a client's identity involved with the protocol, whereas the client establishes a session key with the server. In other words, any client C_i can prove that he is a member of the set C by sending his authenticator at the end of the protocol, however the server doesn't know who he is.

3.3 Computational Diffie-Hellman Assumption

A (t, ε)-$\mathsf{CDH}_{g,G}$ attacker, in a finite cyclic group G of prime order q with g as a generator, is a probabilistic machine $\mathcal{B}$ running in time t such that its success probability $\mathsf{Succ}_{g,G}^{\mathsf{cdh}}(\mathcal{B})$, given random elements g^x and g^y to output g^{xy}, is greater than ε. We denote by $\mathsf{Succ}_{g,G}^{\mathsf{cdh}}(t)$ the maximal success probability over every adversaries running within time t. The CDH-Assumption states that $\mathsf{Succ}_{g,G}^{\mathsf{cdh}}(t) \leqq \varepsilon$ for any t/ε not too large.

4. Security Proofs

At first we show that the EAP protocol of Fig. 1. is provably secure in the random oracle model [6]. Random oracles are used to model cryptographic hash functions which produce a random value for each new query. Note that security in the random oracle model is only a heuristic: it does not imply security in the real world [8]. Nevertheless, the random oracle model is a useful tool for validating natural cryptographic constructions. Security proofs in this model prove security against adversaries that are confined to the random oracle world.

Here we assert that the EAP protocol of Fig. 1. distributes session keys that are semantically-secure and provides unilateral authentication of server S. Note that secure unilateral authentication can be easily extended to mutual authentication by adding another authenticator as suggested in [3], [5].

[Theorem 1] **(AKE/UA Security)** Let P be the EAP protocol of Fig. 1., where passwords are chosen from a dictionary of size N and n is the number of clients. For any adversary $\mathcal{A}$ within a polynomial time t, with less than q_s active interactions with the parties (Send-queries), q_p passive eavesdroppings (Execute-queries) and asking q_h (resp., q_g) hash queries to any $\mathcal{H}_k$ (resp., $\mathcal{G}$), $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) \leq 4\varepsilon$ and $\mathsf{Adv}_P^{\mathsf{S-auth}}(\mathcal{A}) \leq \varepsilon$, with ε upper-bounded by

$$\frac{4q_s}{N} + 4nq_g^2 \times \mathsf{Succ}^{\mathsf{cdh}}_{g,G}(t+3\tau_e) + \frac{q_s}{2^{l_1}} + \frac{3(q_s+q_p)^2}{2q} + \frac{(q_s+q_p+q_g+q_h)^2}{2^{l+1}},$$

where l_1 (resp., l) is the output length of $\mathcal{H}_1$ (resp., $\mathcal{G}$) and τ_e denotes the computational time for an exponentiation in G.

This theorem shows that the EAP protocol is secure against off-line dictionary attacks since the advantage of the adversary essentially grows with the ratio of interactions (number of Send-queries) to the number of passwords. Note that it is sufficient to prove the security with regard to one client due to the fact that each client's password has been registered to the server independently and the latter has no common information to be used for all clients (similar discussion can be founded in Section 3.3 of [17]).

Now we prove that the EAP protocol provides client's anonymity against a passive server.

[Theorem 2] The EAP protocol provides client's anonymity against a passive server in an information-theoretic sense.

[Proof 1] Consider server S who follows the protocol honestly, but it is curious about identity of client C_i involved with the EAP protocol. It is obvious that server S cannot get any information about C_i's identity since the X^* has a unique discrete logarithm of g and, with the randomly chosen x, it is the uniform distribution over G. This also implies that the interactions between either C_i or $C_{j\neq i}$ and S are completely independent one another. In addition, even if server S receives the client's authenticator $V_{C_i} = \mathcal{H}_3(C||S||X^*||Y||Z_j||MS)$ at the end of the EAP protocol (in the case of mutual authentication), the X^* doesn't reveal any information about the client's identity from the fact that the probability for all clients to compute MS is equal.

5. Efficiency

In this section we show how much the EAP protocol is efficient compared to the original anonymous PAKE protocol (appeared in Section 3.2 of [21]) in terms of computation costs and communication bandwidth to be required (see Table 1). In general, the number of modular exponentiations is a major factor to evaluate efficiency of a cryptographic protocol because that is the most power-consuming operation. So we count the number of modular exponentiations as computation costs of client C_i and server S. The figures in the parentheses are the remaining number of modular exponentiations after excluding those that are pre-computable. In terms of communication bandwidth, $|\cdot|$ indicates its bit-length and hash denotes hash functions.

With respect to computation costs in the EAP protocol, client C_i (resp., server S) is required to compute 3 (resp., $2n+1$) modular exponentiations. When pre-computation is allowed, the remaining costs of client C_i (resp., server S) are 2 (resp., $2n$) modular exponentiations. One can easily see that the EAP protocol has about 50% reduction from the APAKE protocol in the number of modular exponentiations for both client and server. With respect to communication bandwidth, the EAP pro-

表1 Comparison of anonymous PAKE protocols as for efficiency where n is the number of clients

Protocols	The number of modular exponentiations		Communication bandwidth
	Client C_i	Server S	
APAKE [21]	6 (4)	$4n+2$ $(3n+1)$	$\lvert C\rvert + \lvert S\rvert + (n+1)\lvert\mathsf{hash}\rvert + (n+2)\lvert p\rvert$
EAP	3 (2)	$2n+1$ $(2n)$	$\lvert C\rvert + \lvert S\rvert + (n+1)\lvert\mathsf{hash}\rvert + 2\lvert p\rvert$

tocol requires a bandwidth of $((n+1)|\mathsf{hash}| + 2|p|)$-bits except the length of identities C and S. Let us consider the minimum security parameters recommended in practice ($|p| = 1024$ and $|\mathsf{hash}| = 160$). The gap of communication bandwidths between the EAP and APAKE protocols becomes larger as the number of clients increases.

6. Conclusions

In this paper, we have proposed an efficient anonymous PAKE (for short, EAP) protocol that provides semantic security of session keys and anonymity against a passive server. We also proved its security of the EAP protocol in the random oracle model with the reduction to the computational Diffie-Hellman problem. The EAP protocol significantly improves efficiency in terms of computation costs and communication bandwidth compared to the original anonymous PAKE protocol [21].

As discussed in [21], one of the interesting problems may be to design an anonymous PAKE protocol against an active server who can deviate the protocol by changing messages at its own.

文 献

[1] M. Abdalla and D. Pointcheval. Simple Password-Based Encrypted Key Exchange Protocols. In *Proc. of CT-RSA 2005*, LNCS 3376, pages 191-208. Springer-Verlag, 2005.

[2] M. Bellare, R. Canetti, and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *Proc. of 30th ACM Symposium on Theory of Computing (STOC)*, pages 419-428, ACM, 1998.

[3] E. Bresson, O. Chevassut, and D. Pointcheval. New Security Results on Encrypted Key Exchange. In *Proc. of PKC 2004*, LNCS 2947, pages 145-158. Springer-Verlag, 2004.

[4] S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In *Proc. of IEEE Symposium on Security and Privacy*, pages 72-84, 1992.

[5] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Proc. of EUROCRYPT 2000*, LNCS 1807, pages 139-155. Springer-Verlag, 2000.

[6] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of ACM CCS '93*, pages 62-73, 1993.

[7] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Proc. of CRYPTO '93*, LNCS 773, pages 232-249. Springer-Verlag, 1993.

[8] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. In *Proc. of the 30th ACM Symposium on Theory of Computing (STOC)*, pages 209-218, ACM, 1998.

[9] W. Diffie and M. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, Vol. IT-22(6), pages 644-654, 1976.

[10] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and Authenticated Key Exchange. In *Proc. of Designs, Codes, and Cryptography*, pages 107-125, 1992.

[11] A. Frier, P. Karlton, and P. Kocher. The SSL 3.0 Protocol. Netscape Communication Corp., 1996.

[12] L. Gong. Optimal Authentication Protocols Resistant to Password Guessing Attacks. In *Proc. of the 8th IEEE Computer Security Foundation Workshop*, pages 24-29, 1995.

[13] S. Halevi and H. Krawczyk. Public-Key Cryptography and Password Protocols. *ACM Transactions of Information and System Security*, Vol. 2(3), pages 230-268, February 1999.

[14] `http://grouper.ieee.org/groups/1363/`

[15] IETF (Internet Engineering Task Force). Transport Layer Security (tls) Charter.

[16] H. Krawczyk. SIGMA: the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols. In *Proc. of CRYPTO 2003*, LNCS 2729, pages 400-425. Springer-Verlag, 2003.

[17] V. Kolesnikov and C. Rackoff. Key Exchange Using Passwords and Long Keys. In *Proc. of TCC 2006*, LNCS 3876, pages 100-119. Springer Verlag, March 2006.

[18] T. Lomas, L. Gong, J. Saltzer, and R. Needham. Reducing Risks from Poorly Chosen Keys. In *Proc. of the 12th ACM Symposium on Operating System Principles*, ACM Operating Systems Review, Vol. 23(5), pages 14-18, 1989.

[19] S. Patel. Number Theoretic Attacks on Secure Password Schemes. In *Proc. of IEEE Symposium on Security and Privacy*, IEEE Computer Society, pages 236-247, 1997.

[20] V. Shoup. On Formal Models for Secure Key Exchange. IBM Research Report RZ 3121, 1999.

[21] D. Q. Viet, A. Yamamura, and H. Tanaka. Anonymous Password-Based Authenticated Key Ex-

change. In *Proc. of INDOCRYPT 2005*, LNCS 3797, pages 244-257. Springer-Verlag, 2005.

[22] W. G. Tzeng. Efficient 1-Out-n Oblivious Transfer Schemes. In *Proc. of PKC 2002*, LNCS 2274, pages 159-171. Springer-Verlag, 2002.

[23] S. B. Wilson, D. Johnson, and A. Menezes. Key Agreement Protocols and their Security Analysis. In *Proc. of IMA International Conference on Cryptography and Coding*, December 1997.

[24] Z. Wan and S. Wang. Cryptanalysis of Two Password-Authenticated Key Exchange Protocols. In *Proc. of ACISP 2004*, LNCS 3108, pages 164-175. Springer-Verlag, 2004.