JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 22, 1295-1308 (2006)

Short Paper

Modular Design for Round-Oriented Password Authentication Protocols

CHI-CHAO CHANG* AND TZONELIH HWANG
*Department of Information Management
Chang Jung Christan University
Tainan, 711 Taiwan
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, 701 Taiwan

Password authentication has always been a natural choice for remote access in network applications. Although there have been many protocols of this kind exists today, few have seriously taken into account and exploited the duplex nature of modern communication networks. Given the capability to send and receive messages at the same time, it would be inefficient in terms of communication and computation resources for participants to wait for responses while doing nothing. There have also been a number of researches focus on parallelization of computations in step-oriented protocols. However, these protocols are often highly integrated and no modification is considered secure without rigorously analyzing them again. Thus, system designers who have restrictions and policies set for by their institutions may find that incorporating these protocols a dreadful task. In this paper, we seek to identify the functional modules in password authentication schemes and give a general procedure for generating protocols with these modules. We also give a proof of security for the generalized protocol produced from the procedure. With modular and round-oriented design, we show that flexible infrastructure can be built to provide sound solutions to password authentication in a wide range of hardware/software implementations and computing capabilities.

Keywords: password authentication, network security, authenticated key exchange, key distribution, cryptography, information system

1. INTRODUCTION

Human memorable passwords have been used in user authentication protocols more than any other ways for their simplicity and security. Password schemes are simple because they are easy to implement, and no additional hardware or software components are required other than standard input devices and security algorithms. There is also no need for users to carry physical tokens such as smart cards to keep their secret information. Besides, since passwords are memorized, these schemes are safe from loss, theft and damage. However, password authentication protocols do have their shortcomings. Memorable passwords are often easy to guess because they either have some relation-

Received June 17, 2004; revised November 16, 2004 & April 26, 2005; accepted November 2, 2005.
Communicated by Shiuhpyng Shieh.

ships to our personal information or are susceptible to dictionary attacks [1]. A good password authentication scheme should be resistant to these attacks or at least detects them within the first few attempts.

Although many researches have been devoted to give secure password authentication protocols. The analysis of protocols has also made a great deal of progress from analyzing against specific attacks to the formal proofs of their security. However, these attempts to provide the single most efficient protocol face challenges not from their security, but from their practicality and flexibility. Each environment is unique in its hardware/software supports, corporate policies, user preferences and authoritative hierarchies. There is simply no single solution to satisfy all password authentication requirements. In this paper, we take an alternative approach by identifying major components in password authentication schemes which can be reorganized into round-oriented protocols and give a general procedure to create secure protocols without sacrificing communicating efficiency. System designers using the technique provided here should be able to create secure password authentication protocols meeting their specific hardware/software restrictions.

1.1 Round vs. Step Oriented Designs

A "round" is an exchange of messages between the communicating parties in which the exchanged information are of no specific order [5]. If one or more messages can be sent without specific sequence or order, we call the time needed to send and receive these messages a "round". In other words, in the *round-oriented design*, messages involved in the same round are *independent* and should be sent simultaneously. At first glance, it seems that round-oriented designs are more complicated since all messages must be reviewed to determine if any of them can be sent independently. However, once we look at a protocol as a composition of it functional modules, it suddenly becames clear that round-oriented design is in fact much more straightforward than step-oriented designs. Functions in step-oriented protocols tend to be highly integrated mostly for the sake of efficiency. On the other hand, round-oriented protocols seem to have looser structures while preserving their efficiency, as we shall see later in the context.

The main difference between round-oriented and step-oriented design is in the design process rather than their primitive components. On the one hand, step-oriented designs have come a long way from simply combining their functional components (like those in EKE [3], A-EKE [2, 4], SPEKE [6] and B-SPEKE [7]) to highly integrated and intricate structures which not even the slightest details can be modified without rigorously reconsider their security. Although these coherent protocols are becoming more and more efficient, they are also harder to be applied if the conditions were in any way different from their original designs. In addition, recent analysis [12] showed that some of these protocols may be vulnerable to new attacks.

Round-oriented designs, on the other hand, are made up with primitive modules which have been known to security communities for some time and are considered safe. Of course, there is no guarantee that combinations of secure components always result in secure protocols and putting the modules together may be a delicate process by itself. Yet with their loose structures, round-oriented designs are easier to verify in terms of their security.

2. SECURITY IN USING THE FRAMEWORK

In this section, we prove the security of the generalized password authenticated key exchange protocol and provide a guideline with which efficient scheme can be generated using the components in the previous section.

2.1 Security Proofs for the Framework

In order to prove the security of the proposed framework, we first define a secure password authenticated key exchange scheme and its components. Then we describe the combining function and prove that the resulting scheme satisfies the security requirements. For the remainder of this section, we assume that $\mathcal{P}$ is the set of all passwords which can be exhaustively searched in time polynomial to the security parameter κ. $\mathcal{R}$ is a random oracle which returns a random bit string of length κ when queried. Exhaustive search in the space of $\{0, 1\}^{\kappa}$, however, cannot be done in polynomial time.

First, we give our definitions of a secure password authenticated key exchange scheme:

Definition 1 Secure Password Authenticated Key Exchange Scheme On input a secret password $\pi \in \mathcal{P}$ and a random tape r, a password authenticated key exchange protocol returns a tuple (α, β, k) where $\alpha, \beta \in \{0, 1\}$ and $k \in \{0, 1\}^{\kappa}$. A *secure* password authenticated key exchange scheme Π must satisfy the following requirements:

Authentication If $\alpha = 1$, then there is an overwhelming possibility that the opponent knows π. Otherwise, the protocol fails.

Key Consistency If $\beta = 1$, then there is an overwhelming possibility that the opponent knows k. Otherwise, the protocol fails too.

Privacy The transcript of the protocol must not reveal information about π and k so that any probability polynomial time (PPT) adversary $\mathcal{A}$ can distinguish π and k from those taken randomly from $\mathcal{P}$ and $\{0, 1\}^{\kappa}$.

Note that we only deal with unilateral security associations in Definition 1. Bilateral security can be achieved when both participants execute the protocol simultaneously. Although bilateral services provide better confidence for both participants, unilateral security associations may be required when one of the participants has limited computing power.

Next, we define the components for the framework before proving that the combination of these components does not reduce the strength of individual modules.

Definition 2 Secure Password Authentication Module (P) On input of a secret password $\pi \in \mathcal{P}$ and a random bit string $r \in \{0, 1\}^{\kappa}$, a secure password authentication module $\mathbf{P}_{\mathcal{R}}(\pi, r) \rightarrow b$ returns a bit $b \in \{0, 1\}$. If $b = 1$, then there is an overwhelming possibility that the opponent shares the same π with the player. Otherwise, the opponent does not know π. The parameters in the transcript are indistinguishable from those directly taken from r.

Definition 3 Secure Key Generation Module (K) On input a random tape $r \in \{0, 1\}^{\kappa}$, a secure key generation module $\mathbf{K}_{\mathcal{R}}(r) \rightarrow k$ returns a bit string $k \in \{0, 1\}^{\kappa}$ which is distributed uniformly. In addition, k cannot be derived from public transcripts of the protocol with non-negligible possibility.

Definition 4 Secure Key Confirmation Module (C) On input a secret key $k \in \{0, 1\}^{\kappa}$, a secure key confirmation module $\mathbf{C}_{\mathcal{R}}(k) \rightarrow b$ returns a bit $b \in \{0, 1\}$. If $b = 1$, then there is an overwhelming possibility that the opponent shares the same k with the player. Otherwise, the opponent does not know k. The parameters in the transcript are indistinguishable from those directly.

In order to produce a secure password key exchange protocol, we must combine the three components defined above in a fashion that the strength of individual modules are preserved. We now give our main theorem:

Theorem 1 A password authenticated key exchange protocol

$$\Gamma_{\mathcal{R}}(\pi) \rightarrow (\alpha, \beta, k)$$

where $\pi \in \mathcal{P}$, $\alpha, \beta \in \{0, 1\}$ and $k \in \{0, 1\}^{\kappa}$ is secure if

$$\Gamma_{\mathcal{R}} \equiv \{\mathbf{P}_{\mathcal{R}}(\pi, r) \rightarrow \alpha, \mathbf{K}_{\mathcal{R}}(r) \rightarrow k, \mathbf{C}_{\mathcal{R}}(k) \rightarrow \beta\}$$

where $r \in \{0, 1\}^{\kappa}$ is a random bit string queried during the protocol.

Proof: Although it seems straightforward that the combination of **P**, **K** and **C** is sufficient for constructing a secure password authenticated key exchange protocol, the critical factor is how they are linked. The proposed framework uses a random bit string r to establish link between the password authentication module and the key generation module. The generated key k is later fed to the key confirmation module to verify that the opponent has indeed recovered the key correctly. Since $r \in \{0, 1\}^{\kappa}$ is random and k is distributed uniformly in $\{0, 1\}^{\kappa}$, an interleaving attack cannot be successful with non-negligible possibility. In addition to provide critical link between module **P** and **K**, the random bit string r has two major functions in the proposing scheme. First, it is used to prevent off-line guessing attack by mixing with the password π. Second, it also serves as a nonce to guarantee the freshness of the key k. Therefore, we conclude that the protocols generated from the proposing framework provide the strength of an integrated protocol. ❑

In Theorem 1, we successfully dissected a password authenticated key exchange protocol into three smaller modules. We also prove that secure modules can be used to create larger constructions without reducing the strength of individual components. However, the implication of Theorem 1 goes beyond simply breaking a secure protocol into smaller parts. By identifying the critical links between the modules, we now have the tool to generate very efficient protocols. For example, we can arrange the password authentication module parallel to the key generation module, thereby reducing the communication cost.

3. NOTATIONS

In the following context, a set of notations are used to represent the nature of variables and computations: Assume that the security parameter is κ. P is the secret password shared only by the two communicating parties. It is an element in some relatively small space $\mathcal{P}(\kappa)$ in which exhaustive search can be done efficiently. In other words, guessing a password correctly has a non-negligible possibility of, say $\delta(\kappa)$. R_A, $R_B \in \{0, 1\}^\kappa$ are large random number generated by party A and B, respectively. These numbers are considered random and sufficiently long so that guessing any of them correctly has some negligible possibility, say $\varepsilon(\kappa)$. F, F_1 and F_2 are intractable one-way functions that reversing their processes are infeasible. In this paper, let's consider these functions as modular exponentiation operations. H function is a cryptographically strong, collision-free hash function. E_B and D_B are the public encryption key and its corresponding private decryption key, respectively, of participant B. Public or private keys followed by parentheses, $E(M)$ or $D(M)$, means encryptions or decryptions of message M in some public-key cryptosystems. Symmetric key encryption scheme is represented in the form of $K[M]$, in which the message M is encrypted with a symmetric key K. Specific symbols or operations will be explained within context.

4. MODULAR COMPONENTS

There are three major functions in a password authentication protocol in security applications: *password authentication*, *key generation* and *key confirmation*. Password authentication is the process to identify the user and verify his/her knowledge to the password. Key generation process is for both communicating parties to share a secret information which will be used in the following communication session, usually as encryption key. Key confirmation process, however, does not provide additional features to both parties except explicit knowledge that their counterparts have had the correct session key. But this does not imply that omitting key confirmation process will improve the efficiency of the protocol. Although we can judge the successful distribution of session keys by whether messages are decrypted successfully in the session, adding an explicit confirmation process makes the protocol a coherent unit so that the security can be examined thoroughly.

In section 2, we proved that secure password authenticated key exchange protocol can be generated from secure components. In this section, we first describe the notations for symbols and operations used. Then we present the prototypes of components which are used to produce round-efficient protocols. Finally, the efficiency of each components are discussed.

For each of the functions in a protocol, there are corresponding modules to perform the task. We now give the details of these modules that commonly appear in current researches of password authentication protocols. System designers who wish to produce a protocol with this technique must consider the different requirements between the modules and select those suitable for their hardware/software environments.

4.1 PA: Password Authentication Structures

4.1.1 Type I: password-based authentication

Since human-memorable passwords are considered susceptible to guessing attacks, authentication with plain password cannot be safe if any information encrypted by or encrypting the password is later revealed. The primitive form of password-based authentication first appears in EKE [3], where both parties encrypt random information linked by an intractable trapdoor function, or in the case of EKE, public-key encryption scheme. This type of authentication is not suitable for parallelization of computation since the two messages are inherently sequential. The authentication process in SPEKE [6] uses a different approach in which two independent random information are encrypted (in the exponentials) and sent without strict causality.

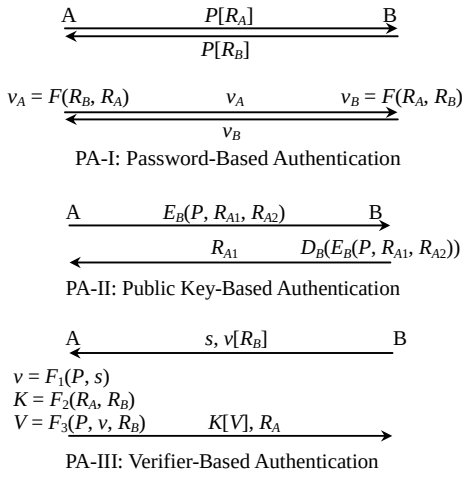


Fig. 1. Password authentication structures.

In this paper, we define the password-based authentication as the form of PA-I in Fig. 1. Two parties, *A* and *B*, first exchange a message of password-encrypted random numbers, $P[R_A]$, $P[R_B]$. Both parties compute the verifier with an intractable function, e.g. $v_a = F(R_A, R_B)$, and send them to their partners for verification.

4.1.2 Type II: public-key-based authentication

This type of password authentication appears in a number of research publications [8-11] and some of them are later attacked and improved. It seems like using a public-key cryptosystem to encrypt passwords for verification is a straightforward and fault-proof approach. However, it is the subtleties that separate good authentication processes from bad ones. First of all, although it is hard to recover passwords by decrypting the ciphertext, it is much easier to guess a password, encrypt with the public key, then compare the result against the captured ciphertext. So the first thing to do is to thwart the

opponent by adding random numbers, called *confounders*, to the encryption. The confounder greatly increases the difficulty for an adversary to perform guessing attacks since she needs to guess both the password and the confounder correctly to match the captured ciphertext.

Depicted in PA-II of Fig. 1 is the basic form of public-key-based authentication. A party, usually the client, uses the server's public key E_B to encrypt his password P and two random numbers R_{A1} and R_{A2} and sends it to the server, who then decrypts the message with its private key D_B and sends one of the random numbers, say R_{A1} back to the client for verification. R_{A2}, the other random number sent encrypted by the client, works as a *confounder* and should be erased right away.

4.1.3 Type III: verifier-based authentication

Verifier-based authentication first appears in A-EKE [4] to provide additional security even after an adversary has access to the server's secret files. In this scenario, the adversary tries to assume the rights of legitimate users with the secret information stored on the server. The key to this type of authentication is to differentiate the knowledge of *real* passwords from their derivatives. In original A-EKE scheme, a client can initiate this process since the verifier can be directly derived from password alone. However, other verifier-based protocols in recent publications require random *salts* to prevent same passwords be transformed into same derivatives, which makes the authenticating process server-initiative.

We adopt the server-initiative structure (PA-III in Fig. 1) here since the model from A-EKE can be seen as a variation of module PA-I. The process begins by server sending the stored random salt s and a random number R_B encrypted with the verifier v of the client. Upon receiving the salt, the client first computes the verifier v with her password and the salt. The verifier is used to recover R_B and generate a secret key K with another random number R_A. Then an *authenticator* V is produced with the password P, the verifier v and R_B. Finally, the secret key K is used to encrypt the authenticator V and sent back to the server along with R_A.

4.2 KG: Key Generation Structures

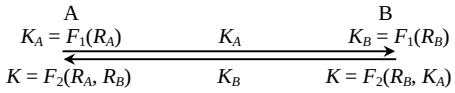
A successful key generation process produces a correct session key with random values contributed by both parties. Note that the key generating process alone does not guarantee that the other party which shares the session key is the intended one. Thus, there must be some relationships between the PA and KG modules in a protocol to link the key generating processes with authentication processes.

4.2.1 Type I: key agreement

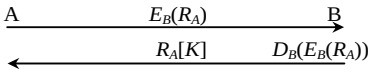
Key agreement process is for both sides to make equal contribution to the generation of session key. This is a fair generating scheme since the session keys are not controlled by a single party, thus making the session key unpredictable even if the random generating algorithm in one of the participants is compromised. The other advantage of this structure is its efficiency. Since both participants generate their own key shares independently, they can be exchanged in a single round. Obviously, the most widely rec-

ognized key agreement scheme is the Diffie-Hellman key exchange protocol. However, there are other intractable functions which are qualified for secure key agreement protocols. For example, if we use a public-key encryption/decryption scheme to send and receive random information and combine it with simple operations, like XOR, this variation may not be very efficient because we assume that both parties have access to the correct public-keys of their opponents. However, it is still a safe key exchange protocol satisfying all the requirements as a KG module.

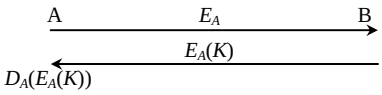
In KG-I of Fig. 2, both participants create their key shares K_A, K_B with an intractable function $F_1()$ and their random numbers R_A, R_B. The received key shares are then fed to another intractable function $F_2()$ along with their own random numbers to produce the session key.



KG-I: Key Agreement



KG-IIa: Key Transfer (Subtype a)



KG-IIb: Key Transfer (Subtype b)

Fig. 2. Key generation structures.

4.2.2 Type II: key transfer

A key transfer scheme put the responsibility to generate the session key on one participant rather than contributed by both. This type of KG protocol may be unfair, but this feature can be preferred or needed under certain circumstances. For example, communicating parties within a hierarchical authoritative regime in which high-ranking participants demand the power to control the generation of session keys. Or, there may be concerns about whether the random generating algorithms is manipulated or compromised when users are logging in from untrustworthy computers or publicly available kiosks. In addition, it is often considered that server-generated session keys are better since they are most likely to be heavily guarded and they might have extra computing power to produce "better" random numbers.

Despite the unbalanced contribution to the session key, players who do not generate session keys are not left undefended. They are protected from replay attacks by selecting random numbers or secret public keys to encrypt the session keys from their partners. To securely transfer the generated session key, it is essential that at least one round of the transferring process be protected by public key schemes. By the origin of the public keys, we divide this type of KG structures into two subtypes depicted as KG-IIa and KG-IIb in Fig. 2:

KG-IIa The initiative participant generates a random number R_A and encrypts it with the partner's public key E_B. R_A is then used by its partner to encrypt the session key K and sent back. The initiator must have access to its partner's correct public key (perhaps through verifying its certificate in a PKI). This scheme is preferable if a client device has limited exponential computing power because we can ask the server to choose smaller public key to facilitate the computation. Another situation this scheme is useful is when hardware support for standard public key encryption/decryption functions (e.g. RSA, El-Gamel) is available since the client needs only symmetric encryption/decryption capability.

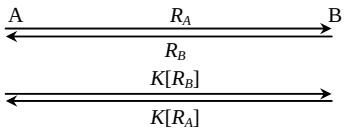
KG-IIb When the participants cannot verify the partner's public key, this type of key transfer must be used. The initiator selects a fresh pair of secret public/private keys E_A and D_A. It sends the public key to its partner who sends back the session key encrypted using this public key. Although this scheme seems susceptible to replay or man-in the middle attacks, the key confirmation module ensures its freshness and the bonding with PA module prevents the messages being manipulated. This scheme, of course, is useful only when the initiator has enough computing power to perform public-key encryptions/decryptions and safely generate key pairs.

4.3 KC: Key Confirmation Structures

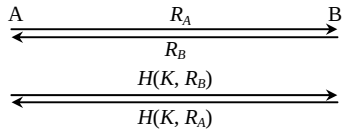
A key confirmation module assures the participants that their partners have had the correct session key. There are two types of key confirmation structures:

4.3.1 Type I: challenge/response

When challenge/response scheme is selected, both participants send random numbers to their partner. These random numbers are then encrypted and sent back for verification as shown in KC-I of Fig. 3. If the deciphered data match the random numbers it has sent in the previous round, then the player considers key distribution a success. In this case, the random numbers serve not only as a key to encrypt the session key, but also as a nonce to ensure the freshness of it.



KC-I: Challenge/Response



KC-II: Hash Image Exchange

Fig. 3. Key confirmation structures.

4.3.2 Type II: hash image exchange

Using hash image exchange schemes are in many ways as secure and efficient as challenge/response ones. Illustrated as KC-II of Fig. 3, both parties send random numbers to their partners who then return the hash images of the random number received along with the session key. It is safe because these cryptographically strong one-way functions cannot be reversed and hash functions are often considered faster than symmetric encryption/decryption functions. However, this type of key confirmation may pose a threat of Denning-Sacco attack (or Known-Key attack) if the following communications use the session key directly. Once an obsolete session key is not erased immediately and an adversary acquires it, it can be used for comparison against the original confirmation message to expose critical information. In many cases, this could lead to off-line guessing attacks. Thus, if hash image exchange scheme is implemented, the session must be transformed using another one-way function and be erased immediately.

4.4 Efficiency of the Modules

In terms of efficiency, there are two aspects to consider: *communication* and *computation*. Listed in Table 1 are the numbers of steps and the rounds of different modules. It shows that the least number of rounds in any round-oriented designed protocol is two no matter how these modules are arranged. Actually, the minimum number of rounds of the resulting protocols is three or four depending on the modules selected. We shall discuss this issue after the protocol generating procedures.

Table 1. Communication efficiency of the modules.

Modules	Steps	Rounds
PA-I	4	2
PA-II	2	2
PA-III	2	2
KG-I	2	1
KG-II	2	2
KC-I	4	2
KC-II	4	2

Table 2. Computation efficiency of the modules.

Modules	Total		Parallelized	
	Exp.	Enc.[1]	Exp.	Enc.[1]
PA-I	2	2	1	1
PA-II	2	0	2	0
PA-III	2	1	2	1
KG-I	4	0	2	0
KG-II	2	1	2	1
KC-I	0	2	0	1
KC-II	0	2	0	1

Computational efficiency, listed in Table 2, however, is a different story. Messages from different modules can be concatenated into one, thus the number of rounds in a protocol may be higher than all the modules it is made of. Computations, on the other hand, must be done in a linear fashion. This linear property suggests that the lower bounds of exponential computations and private key encryption/hashings are 3 and 2, respectively. This will also be elaborated in the discussion of protocol efficiencies.

5. ANALYSIS OF RESULTING PROTOCOLS

5.1 Security

The modules included in this paper have appeared in many secure password authentication protocols. As readers may find, it is the instantiation that determines the security of these modules. Assuming the intractability of $F()$ functions and cryptographically strong hash functions, modules listed in this paper are secure in some aspects and complementary in others. For example, if modular exponential computations are used to instantiate the intractable functions in KG-I module, the result is Diffie-Hellman key exchange protocol. KG-II modules can also be instantiated as RSA or ElGamal encryptions as well. Similarly, PA and KC modules can be easily verified with common primitives. Once these modules are combined into an authentication protocol, the features of these modules complement each other in a tight integration. PA modules solve the authentication problem of KG modules while KC modules verify the success of KG.

Now we also explain why combinations of these modules do not decrease the security of separate modules: steps 1 and 2 in the protocol generating procedure do nothing more than putting the modules in place, thus having no effect on the security. In step 3, because the random numbers in KG modules are used exactly as random numbers in PA modules and key generating functions are intractable, the security of these modules are almost unaffected except that both modules use the same random numbers. Since both PA and KG modules apply their random numbers to intractable functions, the results are not useful in off-line guessing attacks. However, if same intractable functions are applied to both PA and KG modules in a protocol, then malleability is a concern since they share the same input. This can also be avoided by choosing different intractable functions within a protocol.

5.2 Efficiency Analysis

In the discussion of module efficiency, we have divided the issue into communication and computation efficiencies. The discussion on efficiency of the resulting protocol is reserved to this subsection. In the protocol generating procedures, there is no telling how many rounds the resulting protocol will have. A naive concatenation of the module can result to protocols with up to 6 rounds and 5 parallelized exponential computations. (We consider modular exponential functions dominant factors in terms of computation costs over private key encryptions and hashing functions) This is, of course, not exactly the efficient protocol that we hope for.

There are two things to consider when determining the lower bounds of the number of rounds in the protocols generated from the procedures. To begin with, let's assume that the best case scenario is a 2-round protocol in which all modules are perfectly parallelized. First, there must be an initiatory message sent by one of the parties to begin the process. This adds one round to the resulting protocol in cases when the first round of the completed protocols is two-way or server-initiated. Secondly, stage 2 of the protocol generating procedures demands that KC module be completed at least one round after PA module for security reasons. This restriction makes the lower bounds of the resulting protocols vary with the modules of selection. For example, a protocol with PA-I modules is at least 4 rounds since full parallelization is not feasible. However, protocols with PA-II modules can be squeezed into 3 rounds if we align the first round of PA-II module with the initiatory message. So, the lower bounds of the resulting protocols will be 3 or 4 rounds depending on the selection of modules.

6. CONCLUSIONS

An alternative approach to the problem of creating password authentication protocols has been presented in this paper. Unlike previous publications which aim at providing new or improved protocols, we try to raise the issue to a more abstract level. In this paper, password authentication protocols are considered not as individual entity but as the combination of functional modules. By taking a different perspective, the possibility for a new and flexible design regime emerges. With the procedures given in this paper, engineers can design their own password authentication protocols to meet the hardware, software and policy requirements of their own institutions. Protocols generated according to the procedures are consistent in terms of their security and round-efficiency. If a support infrastructure is built to provide computing capabilities of the primitive modules, users wishing to run password authentication from remote locations may be able to negotiate new protocols with the servers according to the computing power and trustworthiness of the remote systems.

Advantages of round-oriented designed protocols include their efficiency, security and practicality. The resulting protocols are efficient because round-oriented designs exploit the duplex nature of modern communication networks. Parallelized message exchange processes and computations reduce the time a client or server system spend waiting for responses. Also, these protocols are heuristically secure because the relationships between modules have very little impact on the integrity of individual modules, which have been seen in a variety of existing protocols and are considered to be secure. Finally, the protocol generating procedures presented in this paper provide system designers tools to design their own password authentication schemes according to the features and restrictions in the environments the protocols are to be used.

Implications of round-oriented designs in this paper may be more than providing standard procedures in constructing password authentication protocols. We hope that this concept will encourage further exploration on modules and their variations to give new ways to categorize and analyze existing protocols as well as creating new ones.

REFERENCES

1. A. Conklin, G. Dietrich, and D. Walz, "Password-based authentication: a system perspective," in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 2004, pp. 170-179.
2. A. M. Barmawi, S. Takada, and N. Doi, "Augmented encrypted key exchange using RSA encryption," in *Proceedings of the 8th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Vol. 2, 1997, pp. 490-494.
3. S. M. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks," in *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 1992, pp. 72-84.
4. S. M. Bellovin and M. Merritt, "Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993, pp. 244-250.
5. R. Gennaro, "A protocol to achieve independence in constant rounds," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, 2000, pp. 636-647.
6. D. P. Jablon, "Strong password-only authenticated key exchange," in *Proceedings of the ACM SIGCOMM Computer Communication Review*, Vol. 26, 1996, pp. 5-26.
7. D. P. Jablon, "Extended password key exchange protocols immune to dictionary attacks," in *Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, 1997, pp. 248-255.
8. T. Kwon, M. Kang, and J. S. Song, "An adaptable and reliable authentication protocol for communication networks," in *Proceedings of the IEEE INFOCOM '97, The Conference on Computer Communications, 16th Annual Joint Conference of the IEEE Computer and Communications Societies*, 1997, pp. 737-744.
9. T. Kwon and J. S. Song, "Security and efficiency in authentication protocols resistant to password guessing attack," in *Proceedings of the 22nd IEEE Conference on Local Computer Networks* (*LCN*), 1997, pp. 245-252.
10. T. Kwon and J. S. Song, "Efficient key exchange and authentication protocols protecting weak secrets," *IEICE Transactions on Fundamentals*, Vol. E81-A, 1998, pp. 156-163.
11. T. Kwon, M. Kang, S. Jung, and J. Song, "An improvement of the password-based authentication protocol (k1p) on security against replay attacks," *IEICE Transactions on Communications*, Vol. E82-B, 1999, pp. 991-997.
12. M. Zhang, "Analysis of the speke password-authenticated key exchange protocol," *IEEE Communications Letters*, Vol. 8, 2004, pp. 63-65.

Chi-Chao Chang (張智超) received the B.S. degree in Microbiology from Soochow University in 1990, the M.S. degree in Computer Science from State University of New York at Albany in 1992 and the Ph.D. degree from the National Cheng Kung University in 2005. He joined the Chang Jung University since 1993 and is now an assistant professor there. His research interests include information security, mobile agent systems, digital signatures and quantum cryptography.

Tzonelih Hwang (黃宗立) was born in Tainan, Taiwan, in March 1958. He received his undergraduate degree from National Cheng Kung University, Tainan, Taiwan, in 1980, and the M.S. and Ph.D. degrees in Computer Science from the University of Southwestern Louisiana, U.S.A., in 1988. He is presently a professor in Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include cryptology, network security, and coding theory.