A Novel Smart-Card Based Authentication Scheme Using Proactive Secret Sharing

Yuesheng Zhu, Bojun Wang*, Cheng Cai

Lab of Communication and Information Security, Institute of Big Data Technologies, Shenzhen Graduate School, Peking University, Shenzhen, 518052, China.

* Corresponding author. Email: wangbojunhappy@gmail.com
Manuscript submitted October 3, 2014; accepted June 1, 2015.
doi: 10.17706/ijcce.2016.5.3.196-205

Abstract: Traditional smart-card oriented authentication schemes provide a secure method for authorized users to login a remote server through insecure networks. Many modified schemes are presented to enhance the security properties, which include mutual authentication, various attacks resistance, table-free in server, etc. However, the authentication procedures become complicated with abovementioned enhancements, so these enhanced schemes are not practical solutions for the smart card based authentication system. Moreover, since the master key stored in a single server is a sensitive and long-lived secrecy, it is vulnerable to an adversary's agelong attack. Therefore, in this paper, we present an efficient mutual authentication scheme, which provides a practical solution for an industrial application system using smart cards. In addition, our scheme implements the proactive secret sharing method, namely, the master key is divided into several distributed keys, which are separately stored in different servers. Our scheme provides a mechanism to update all distributed keys in every time interval, and detect and then recover the corrupted distributed keys, as well.

Key words: Distributed key, mutual authentication, proactive secret sharing, smart cards.

1. Introduction

T. C. Wu [1] proposed a smart-card oriented authentication scheme based on some properties of Euclidean geometry. This scheme can achieve the login and authentication phase easily. However, M. S. Hwang found the weakness [2] of T. C. Wu's scheme and then improved the security in the authentication process [3] by using ElGamal's public key cryptosystem [4]. Many modified methods [5]-[10] based on Hwang 's scheme were presented to enhance the security.

In 2000, C. K. Chan [11] analyzed the weakness of Hwang's scheme and pointed out that this scheme could not resist the masquerade attack. A different masquerade attack on Hwang's scheme was found by Jau-Ji Shen *et al.* [12], and then they modified the register phase to against this masquerade attack. These schemes above provide a method for the remote server to authenticate a user. Chun-Ta Li [13] proposed a mutual authentication scheme between two communication parties. Yet the complex procedures and a relatively large amount of interactive messages are the open issues of the scheme. We proposed a mutual authentication scheme between the smart card and the authentication server (AS). After the smart card finishing the authentication of AS, the AS can perform the smart card's authentication. The smart card and the AS can access each other's data after the mutual authentication.

Secret sharing scheme is first proposed by Shamir [14], which can be applied to manage important information. Blakley [15], C. Asmuth, and J. Bloom [16] also present different secret sharing methods. Later, many modified secret sharing schemes were presented. The first verifiable secret sharing (VSS) was proposed by B. chor *et al*. [17]. Also, Feldman [18] and Pedersen [19] improved VSS to two different non-interactive VSS schemes. Amir Herzberg *et al.* [20] proposed a proactive secret sharing scheme, in which provides protocols of share renewal, corrupted share detection, and corrupted share recovery. In the smart card based authentication process, another crucial issue is about master key, which is possessed by only one server. However, the master key is invariable information, so it can be leak under the continuous attack of an adversary. Therefore, the traditional proactive secret sharing method can be used to manage the master key.

2. Related Theories

2.1. ElGamal's Public Key Cryptosystem

Submit your manuscript electronically for review. ElGamal's cryptosystem is based on the asymmetric cryptographic algorithm. For example, user A wants to send a message M to user B using ElGamal's cryptosystem. In the cryptosystem, p, g are two public parameters. Here p is a large prime number and $g < p$. x is user B's private key and $y = g^k \mod p$ is user B's public key. To encrypt the message M, A selects a random number k satisfies $\gcd(k, p-1) = 1$ and computes a, b to send to the user B.

$$\begin{aligned} a &= g^k \bmod p \\ b &= y^k M \bmod p \end{aligned} \tag{1}$$

After user B receives a, b, he can get the secret message M by computing $M = b / a^x \pmod p$.

2.2. Cryptographic Tools

Our scheme requires asymmetrical encryption [21] and unforgettable signatures [22]. E.g., a sender S wants to send a message to a receiver R, S using R's public key to encrypt this message is denoted by $Enc_{PU_R}(\text{message})$. In addition, S sends a message with his or her signature using S's private key. This signature is denoted by $Sig_{PR_S}(\text{message})$.

3. A Smart Card Based Mutual Authentication Scheme

3.1. Derived Keys of Smart Cards

Any attacker can thoroughly analyze a smart card since it can be easily got [23]. If the master key is stored in a smart card, the attacker can access the smart card's content and read out the master key. Therefore, a derived key should be present in the smart card instead of the master key. The derived key is generated using a cryptographic algorithm. The smart card's number and the smart key are inputted in this algorithm as parameters. The derived key is unique and can be used to identify the smart card in the entire system, because the smart card's number is unique when the card is manufactured. The derived key can be got by the following function.

$$\text{Derived key} = \text{Enc (card number, master key)} \tag{2}$$

For example, in registration phase, the AS computes the derived keys K_i of each card i, using the function above to encrypt the information of card number ID_i and its master key K_s. The derived key stored in the

card i is $K_i = ID_i^{K_s}$. It is stored as a security file in the smart card. Any device cannot read it before smart card authenticates the remote server successfully. The derived key stored in the card is used to lock data in the card and serves to complete the mutual authentication process with the remote server.

3.2. Protocol of Smart Card Based Mutual Authentication Scheme

Before the communication between the smart card and the remote server, the mutual authentication has to be performed between the two parties. The smart card authenticates the server fist to confirm it is a legal server. After that, the remote server requests the AS to authenticate smart card.

In traditional authentication scheme, master key is possessed by AS. Yet the master key can be leak under malicious attack or be lost when the AS shuts down. In our authentication scheme, master key is stored neither in smart cards nor the AS, so an attacker cannot get the master key by attacking smart cards and the AS. Our authentication scheme implements the proactive secret sharing method [20]. The master key is split into several pieces (distributed keys) using Shamir's secret sharing method [14], and each sub-server has only one distributed key. Once there is a request from a smart card, the AS will start the master key recovery process. The sub-servers send their distributed keys and reconstruct the original master key. In addition, the AS will manage the renewal process of distributed keys and the detection and recovery process of corrupted distributed keys.

We introduce our new authentication scheme using smart cards in this section, which has the feature of mutual authentication and can resist the masquerade attack. The security of our mutual authentication is based on ElGamal's public key cryptosystem and intractability of discrete log problem. To make our scheme be easily understood, we introduce proactive secret sharing method of AS in Section 4.

1) Registration phase (see Fig. 1)

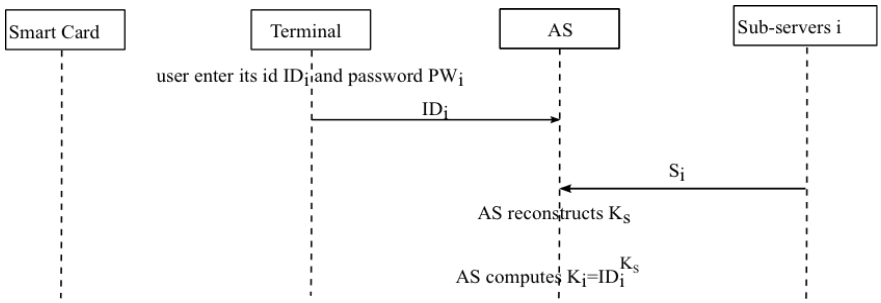


Fig. 1. Registration process.

A smart card is registered in an AS before it is assigned to a user. Namely, the AS uses its master key to generate a derived key and stores this key as a security file in the smart card. In the registration and authentication processes, the interaction between smart cards and AS relies on a terminal device, which is an interactive medium between the smart card and the AS. The user just needs to insert the smart card into a terminal device.

We present our protocol in the registration phase and the authentication phase. The notations used in our scheme are listed in the Table 1.

Step 1: If a user U_i wants to register his or her card in AS, he or she has to input the password in AS's terminal. Then the terminal reads the card number ID_i and then sends ID_i, PW_i to AS.

Step 2: The AS uses its master key K_s to compute U_i's derived key $K_i = ID_i^{K_s} \bmod p$ and stores p, $h(g)$, $h(PW_i)$ in the smart card. In addition, K_i is stored as a security file in the smart card. It locks the data p, $h(g)$, $h(PW_i)$ with a program, so any device cannot access the locked data before be successful authenticated by the smart card.

Table 1. Notations in Our Scheme

U_i	User
AS	Authentication server's name
ID_i	Smart card's number
PW_i	User i's password
K_i	Derived key of a smart card
K_s	Master key of an authentication server
R_i	A random number generated by a smart card
R_s	A random number generated by an AS
p	A large prime number
$h(\sqcup)$	One-way hash function
$+$	XOR operation
N_i	A nonce in i round
K_{CS}	A session key used for message transmission
SID	Session ID

2) Authentication phase

Before the communication between the smart card and the remote server, the mutual authentication has to be performed between the two parties. The smart card authenticates the server fist to confirm it is a legal server. If this authentication is successful, then the remote server requests AS to authenticate smart card. We describe the mutual authentication between the smart card and the server, respectively.

- Smart Card authenticates AS (see Fig. 2)

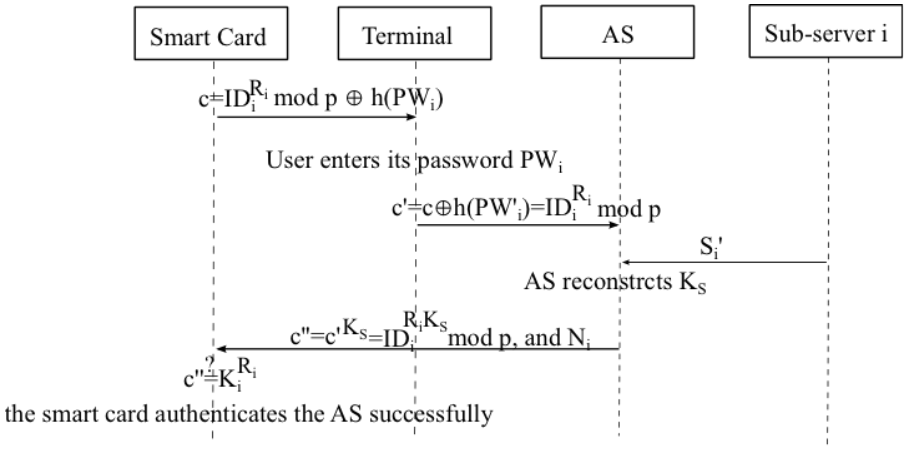


Fig. 2. The smart card authenticates the AS.

Step 1: The smart card generates a random number R_i in $GF(p)$ and sends the challenge $c=(ID_i^{R_i} \bmod p)\oplus h(PW_i)$ to AS through the terminal. Here R_i satisfies $\gcd(R_i, p-1)=1$. Then the user input his or her password in the terminal.

Step 2: The terminal sends a challenge $c'=c\oplus h(PW_i)=ID_i^{R_i} \bmod p$ to AS. Then the AS computes $r=(c')^{K_s}=ID_i^{R_iK_s} \bmod p$ and sends r and a nonce N_i to the smart card through the terminal.

Step 3: The smart card computes $K_i^{R_i}$ to verify weather $K_i^{R_i}=ID_i^{R_iK_s}=(c')^{K_s}$. If it holds, the smart card authenticates the AS successfully.

- AS authenticates Smart card (see Fig. 3)

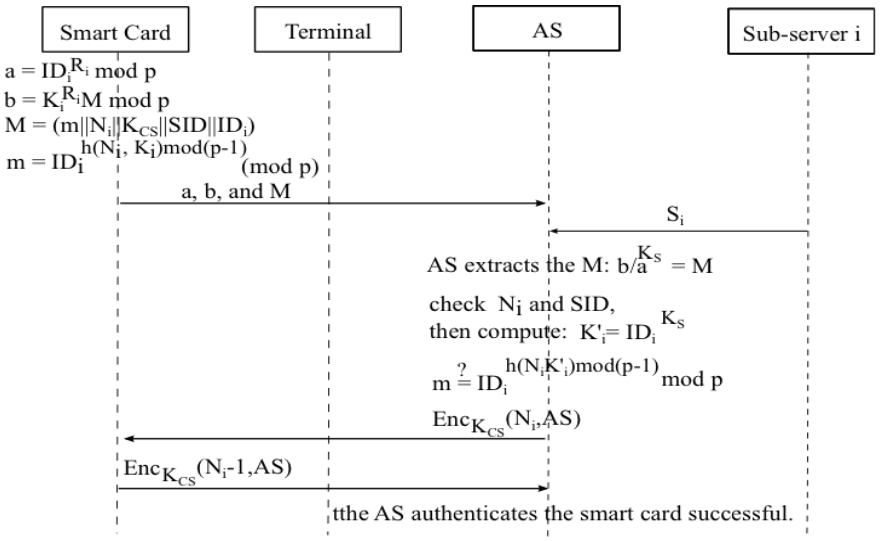


Fig. 3. The AS authenticates the smart card.

After the smart card successfully authenticates the AS, the AS can access the data p, $h(g)$, $h(PW_i)$. The data can be used to authenticate the smart card.

Step 1: The smart card generates a random number R_i in $GF(p)$ and sends the messages a, b, M to the AS.

$$\begin{aligned} a &= ID_i^{R_i} \bmod p \\ b &= K_i^{R_i} M \bmod p \\ M &= (m \| N_i \| K_{CS} \| SID \| ID_i) \\ m &= ID_i^{h(N_i \oplus K_i) \bmod (p-1)} \bmod p \end{aligned} \tag{3}$$

Here, M is the secret message that the smart card wants to send to the AS for authentication purpose.

Step 2: The AS computes $M = b / a^{K_s}$ and then check weather N_i, SID are fresh. If they are fresh, then the AS computes $K_i' = ID_i^{K_s}$ and then checks weather $m = ID_i^{h(N_i \oplus K_i') \bmod (p-1)} \bmod p$.

Step 3: If step 2 holds, the ASsends $Enc_{K_{CS}}(N_i, AS)$ to the smart card. Then the smart card replies a message $Enc_{K_{CS}}(N_i - 1, AS)$. At this time, the AS authenticates the smart card successfully.

4. Distributed Keys Based on Proactive Secret Sharing

4.1. Master Key Management

In the above authentication process, the AS's master key is crucial. That is because once the master key is leaked to an attacker, the attacker can use the master key to fabricate an illegal AS. Since the illegal AS possesses the correct master key, it can successfully pass the authentication of smart cards. This is really dangerous for smart cards because it can be access by the illegal AS. Since the smart key is sensitive information, a good solution to protect such significant secrecy is splitting it and storing it in different sub-servers. Therefore, we apply secret sharing method to the master key management. Namely, using the (k, n)-threshold secret sharing scheme to split the master key into n pieces (distributed keys) and stored these distributed keys in n different sub-servers. Any k sub-servers can reconstruct the original master key and use it to authenticate the smart cards. It is more difficult for an attacker to get k distributed keys (secret shares) to reconstruct the original master key.

In the initialization phase, the system constructs a random polynomial f of degree at most k-1 with coefficient in, $f(x)=\alpha_0+\alpha_1 x+\alpha_2 x^2+...+\alpha_{k-1}x^{k-1} \bmod q$, in which $\alpha_0=K_s$. The master key K_s is divided into n pieces $S_1,S_2,...,S_n$ and distributed to n sub-servers $P_1,P_2,...,P_n$, respectively. There are nsub-servers $P=\{P_1,P_2,...,P_n\}$ share a master key K_s based on Shamir's (k, n)-threshold secret sharing scheme. An attacker cannot learn the master key K_s even if he or she knows k-1 distributed keys, and any k of the sub-servers can still reconstruct the original master key even if n-k sub-servers are corrupted because of the attackers or the sub-servers' normally shutdown.

4.2. Periodic Distributed Keys Renewal Protocol

Using threshold secret sharing to manage master key improves the security of master key, because an attacker has to get k distributed key. However, the master key is inherently long-lived secrecy, so the attacker can spend his or her whole lifetime collecting the distributed key. As a result, the traditional secret sharing scheme is not sufficient.

In our scheme, the sub-servers periodically refresh their distributed keys. The entire system time is divided into a series of time periods. At the beginning of every time period, the AS requests for the renewal of distributed keys. All sub-servers then update their distributed keys according to the periodic distributed key renewal protocol. For example, a weekly renewal of distributed keys can reduce the time for the attacker to get k distributed keys from entire lifetime to one week. In addition, the attacker needs to destroy n-k distributed keys in a single time interval.

The communication among sub-servers is multicast group communication. That is, each sub-server broadcasts information to all the sub-servers in the group. The protocol of periodic distributed key renewal is illustrated as follows.

Step 1: In period t-1, the master key K_s is split into the distributed keys with the polynomial $f^{(t-1)}(x)$. In the period t, every sub-server P_i chooses a random polynomial $G_i^{(t)}(x)=a_{i,1}{}^{(t)}x+a_{i,2}{}^{(t)}x^2+...+a_{i,(k-1)}{}^{(t)}x^{k-1}$ mod q, such that $G_i^{(t)}(0)=0$.

Step 2: Every sub-server P_i computes $u_{ij}^{(t)}=G_i^{(t)}(j),1\le i,j\le n,i\ne j$ and sends $Enc_{PU_j}(u_{ij}^{(t)})$ to sub-servers $P_j,1\le i,j\le n,i\ne j$ respectively. Meanwhile, the sub-server broadcasts the poof $C_j=g^{a_{i,w}{}^{(t)}} \bmod q, w=1,2,...,k-1$. The receiver P_j can verify the correctness of the renewal information by checking $g^{u_{ij}^{(t)}}=\Pi_{w=1}^{k-1}(C_w)^{j^w}$.

Step 3: After every sub-server P_i receives all the other renewal information $u_{ij}{}^{(t)},1\le i,j\le n,i\ne j$, the sub-server P_i needs verify the validity of the renewal information as described above. If the verification succeeds, the sub-server P_i broadcasts the acceptance message represents that all the renewal information is valid.

Step 4: If the sub-server P_i receives all the other sub-servers' acceptance messages, P_i computes his new distributed key as Eq. 4 and erases the previous distributed key $S_i^{(t-1)}$.

$$S_i^{(t)}=S_i^{(t-1)}+u_{1i}{}^{(t)}+u_{2i}{}^{(t)}+...+u_{ni}{}^{(t)} \bmod q \tag{4}$$

Analysis: k updated distributed keys can be used to reconstruct the polynomial $f^{(t-1)}(x)+G_1^{(t)}(x)+G_2{}^{(t)}(x)+...+G_n{}^{(t)}(x)$. The master key does not change, because of the Eq. 5.

$$f^{(t-1)}(0)+G_1^{(t)}(0)+G_2^{(t)}(0)+\ldots+G_n^{(t)}(0) = f^{(t-1)}(0)+0+\ldots+0=f^{(t-1)}(0)=K_s \quad (5)$$

4.3. Detection of Corrupted Distributed Keys

A corruption of a certain sub-server can be easily detected in some cases. That is because a corrupted key can be detected by the verification process of the distributed key renewal phase. However, after the update phase, once an attacker successfully attacks a sub-server, he or she can modify the distributed key stored in this sub-server or change server's behavior. Also, it is possible that a sub-server has normal severe problem (e.g., disconnection, power failures etc.). All the situations above will cause distributed key's corruption. Therefore, it is necessary to periodically detect corrupted distributed keys. The protocol of corrupted distributed keys detection is introduced as follows.

Step 1: Each sub-server P_i stores $y_j^{(t)}=g^{S_j^{(t)}} \bmod p,\ j\in(1\ldots n)$. In the initial period, each sub-server stores $y_j^{(0)}=g^{S_j^{(0)}},\ j\in(1\ldots n)$.

Step 2: In the update phase, each sub-server P_i updates its set $\{y_j^{(t)}\},\ j=(1\ldots n)$ by computing Eq. 6. Set B_i represents the set of sub-server that sends incorrect renewal information.

$$y_j^{(t)}=y_j^{(t-1)}*g^{u_{1j}^{(t-1)}}*g^{u_{2j}^{(t-1)}}*\cdots g^{u_{kj}^{(t-1)}}*\cdots*g^{u_{nj}^{(t-1)}},\ k\notin B_i \quad (6)$$

Step 3: Each sub-server broadcasts the value of the set $\{y_j^{(t)}\},\ j=(1\ldots n)$ to the other sub-servers, together with its signature $Sig_{PR_i}(y_j^{(t)}),\ j=(1\ldots n)$ on these values.

Step 4: After receiving all the messages from the other sub-servers, each sub-server checks the signature on these values. For each sub-server P_i, according to the received messages, it can update its set B_i by majority principle. That means every sub-server can judge which sub-server holds a corrupted distributed key.

4.4. Recovery of Corrupted Distributed Keys

If corrupted sub-servers in set B hold corrupted distributed keys, the other non-faulty sub-servers in set $D = A - B$ have to implement the recovery protocol to recover the correct distributed keys. A straightforward way to recover the corrupted distributed keys $S_r=f^{(t)}(r),\ r\in B$ is each sub-server in set D sending its distributed keys to P_r. However, this method could expose the master key to sub-server P_r, because P_r could collect more than k distributed keys and use them to reconstruct the master key. Therefore, the recovery protocol is presented as follows, in which each sub-server in set D generates a new distributed key and then sends it to P_r. P_r can use these new distributed keys to recover its distributed key without learning about the original distributed keys $\{S_i\},\ i\in D$. The detail of the recovery protocol is illustrated as follows.

Step 1: Each sub-server $P_i, i\in D$ picks a random polynomial δ of degree at most k-1 with coefficient in $\delta_i(x)=\delta_{i,0}+\delta_{i,1}x+\delta_{i,2}x^2+\ldots+\delta_{i,k-1}x^{k-1} \bmod q$ $\delta_i(x)=\delta_{i,0}+\delta_{i,1}x+\delta_{i,2}x^2+\ldots+\delta_{i,k-1}x^{k-1} \bmod q$ such that $\delta_i(r)=0$ and $\delta_{i,0}=-(\delta_{i,1}r+\delta_{i,2}r^2+\ldots+\delta_{i,k-1}r^{k-1}) \bmod q=-\sum_{j\in\{1\ldots k\}}\delta_{i,j}r^j \pmod q$.

Step 2: Each sub-server $P_i, i\in D$ broadcasts $\{Enc_{PU_j}(\delta_i(j))\}\ j\in D$.

Step 3:Each sub-server $P_i, i\in D$ creates its new distributed key by computing $S_i^{'}=S_i+\sum_{j\in D}\delta_j(i)$ and

then sending $Enc_{PU_r}(S_i^{'}), r \in B$ to $P_r, r \in B$.

Step 4: $P_r, r \in B$ decrypts these new distributed keys and uses them to recover its original distributed key S_r.

Analysis: The new distributed keys are as follows.

$$S_i^{'} = S_i + \sum_{j \in D} \delta_j(i) = S_i + \delta_1(i) + \delta_2(i) + ... + \delta_k(i) + ... + \delta_n(i), k \in D \tag{7}$$

Therefore, P_r can use these new distributed keys to reconstruct the polynomial $F(x) = f(x) + \delta_1(x) + \delta_2(x) + ... + \delta_k(x) + ... + \delta_n(x), k \in D$, so P_r can successfully recover its original distributed key S_r by computing Eq. 8.

$$F(r) = f(r) + \delta_1(r) + \delta_2(r) + ... + \delta_k(r) + ... + \delta_n(r), k \in D = f(r) + 0 + ... + 0 = f(r) = S_r \tag{8}$$

4.5. Master Key Reconstruction

When the AS authenticates a smart card, the master key has to be reconstructed. Each distributed key stored in a sub-server is sent to the AS and the master key is reconstructed in AS.

After all sub-servers have sent the distributed keys, the AS can verify these submitted keys by $y_j^{(t)} = g^{S_j^{(t)}} \bmod p,\ j \in (1...n)$ in the phase of corrupted key detection. After that, the AS can choose k correct distributed keys to reconstruct the master key K_s by Eq. 9.

$$\begin{aligned} f(x) &= \sum_{j=1}^{k} \left(\prod_{l \neq j} \frac{x - i_j}{i_l - i_j} \right) f(i_j) \\ K_s &= f(0) = \sum_{j=1}^{k} \left(\prod_{l \neq j} \frac{i_j}{i_j - i_l} \right) f(i_j) \end{aligned} \tag{9}$$

5. Conclusion

In this paper, we have introduced our efficient mutual authentication scheme for smart cards. The specific management of the master key using proactive secret sharing method has also been presented. The master key is reconstructed when a smart card requests to access the remote server. In addition, the distributed keys are periodically updated so that it is more difficult for an adversary to get enough distributed keys in one-time interval. The protocols in our scheme can also detect the corrupted distributed keys and then recover them.

References

[1] Wu, T. C. (1995). Remote login authentication scheme based on a geometric approach. *Computer Communications, 18(12),* 959–963.

[2] Hwang, M. S. (1999). Cryptanalysis of a remote login authentication scheme. *Computer Communications. 22(8),* 742–744.

[3] Hwang, M. S., & Li, L.-H. (2000). A new remote user authentication scheme using smart cards. *IEEE Transactions on Consumer Electronics, 46(1),* 28–30.

[4] ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *Advances in Cryptology,* 10–18.

[5] Lee, C. C., Hwang, M. S., & Yang, W. P. (2002). A flexible remote user authentication scheme using smart cards. *ACM SIGOPS Operating Systems Review, 36(3),* 46–52.

[6] Lee, C. C., Li, L. H., & Hwang, M. S. (2002). A remote user authentication scheme using hash functions. *ACM SIGOPS Operating Systems Review*, *36(4)*, 23–29.

[7] Li, L. H., Lin, I. C., & Hwang, M. S. (2001). A remote password authentication scheme for multiserver architecture using neural networks. *IEEE Transactions on Neural Networks, 12(6),* 1498–1504.

[8] Hwang, M. S., Lee, C. C., & Tang, Y. L. (2002). A simple remote user authentication scheme. *Mathematical and Computer Modeling, 36(1-2),* 103–107.

[9] Peyravian, M., & Zunic, N. (2000). Methods for protecting password trans mission. *Computers & Security, 19(5),* 466–469.

[10] Manber, U. (1996). A simple scheme to make passwords based on one-way functions much harder to crack. *Computers & Security, 15(2),* 171–176.

[11] Chan, C. K., & Cheng, L. M. (Nov. 2000). Cryptanalysis of a remote user authentication scheme using smart cards. *IEEE Transactions on Consumer Electronics, 46(4),* 992–993.

[12] Shen, J. J., Lin, C. W., & Hwang M. S. (2003). A modified remote user authentication scheme using smart cards. *IEEE Transactions on Consumer Electronics, 49(2),* 414–416.

[13] Li, C. T., & Hwang, M. S. (2010). An efficient biometrics-based remote user authentication scheme using smart cards. *Journal of Network and Computer Applications, 33(1),* 1–5.

[14] Shamir, A. (1979). How to share a secret. *Communications of the ACM, 22(11),* 612–613.

[15] Blakley, G. R. (1979). Safeguarding cryptographic keys. *Proceedings of Managing Requirements Knowledge, Int. Workshop* (pp. 313–317*).* Los Alamitos, CA, USA.

[16] Asmuth, C., & Bloom, J. (Mar. 1983). A modular approach to key safeguarding. *IEEE Trans. Inf. Theory, 29(2),* 208–210.

[17] Chor, B., Goldwasser, S., Micali, S., Awerbuch, B. (Oct. 1985). Verifiable secret sharing and achieving simultaneity in the presence of faults. *Proceedings of 26th Annu. Symp. on Foundations of Computer Science* (pp. 383–395). Portland, OR, USA.

[18] Feldman, P. (Oct. 1987). A practical scheme for non-interactive verifiable secret sharing. *Proceedings of 28th Annu. Symp. on Foundations of Computer Science* (pp. 427–438). Los Angeles, CA, USA.

[19] Pedersen, T. P. (1992). Non-interactive and information-theoretic secure verifiable secret sharing, *Proceedings of the 11th Annu. Int. Cryptology Conf. on Advances in Cryptology, ser. CRYPTO '91* (pp. 129–140). London, UK.

[20] Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M. (1995). Proactive secret sharing or: How to cope with perpetual leakage. *Advances in Cryptology*, 339-352.

[21] Goldwksser, S., & Micali, S. (1984). Probabilistic encryption. *Journal of Computer and System Sciences, 28(2),* 270-299.

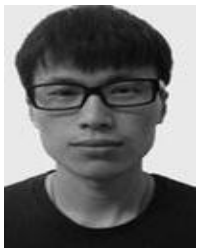
[22] Goldwasser, S., Micali, S., & Rivest, R. (1988). A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing, 17(2), 281-308*.

[23] Rankl, W., & Effing, W. (2010). *Smart Card Handbook*. John Wiley & Sons.

Yuesheng Zhu was born in 1959. He received his B.Eng. degree in radio engineering, M.Eng. degree in circuits and systems and Ph.D. degree in electronics engineering in 1982, 1989, and 1996, respectively. He is currently working as a professor at the Key Labs of Communication and Information Security, School of Electronic and Computer Engineering, Peking University. He is a senior member of IEEE, fellow of China Institute of Electronics, and senior member of China Institute of Communications. His research

interests include communication and information security, digital signal processing, multimedia technology.

Bojun Wang was born in Jiangsu province of China, in 1990. She received the B.S. degree in networking engineering from College of Computer Science, Chongqing University, Chongqing, China. She is currently pursuing the M.S. degree in computer application technology at the Lab of Communication and Information Security, Shenzhen Graduate School, Peking University. Her research interests include information security and its applications, secret sharing and game theory.



Cheng Cai was born in 1989. He received the B.S. degree in computer science from the College of Computer Science, Beijing University of Technology, Beijing, China. He is currently pursuing the M.S. degree in computer application technology at the Lab of Communication and Information Security, Shenzhen Graduate School, Peking University. His research interests include cryptography protocol, information security and network security.