Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VI (2011), No. 2 (June), pp. 227-235

A Time-Bound Ticket-Based Mutual Authentication Scheme for Cloud Computing

Z. Hao, S. Zhong, N. Yu

Zhuo Hao
1. University of Science and Technology of China
Department of Electronic Engineering and Information Science
Hefei, Anhui 230027, P.R.China, and
2. State University of New York at Buffalo
Department of Computer Science and Engineering
201 Bell Hall, Amherst, NY 14260, USA
E-mail: hzhuo@mail.ustc.edu.cn

Sheng Zhong
State University of New York at Buffalo
Department of Computer Science and Engineering
201 Bell Hall, Amherst, NY 14260, USA
E-mail: szhong@buffalo.edu

Nenghai Yu
University of Science and Technology of China
Department of Electronic Engineering and Information Science
Hefei, Anhui 230027, P.R.China
E-mail: ynh@ustc.edu.cn

Abstract: Cloud computing is becoming popular quickly. In cloud computing, people store their important data in the cloud, which makes it important to ensure the data integrity and availability. Remote data integrity checking enables the client to perform data integrity verification without access to the complete file. This service brings convenience to clients, but degrades the server's performance severely. Proper schemes must be designed to reduce the performance degradation.

In this paper, a time-bound ticket-based mutual authentication scheme is proposed for solving this problem. The proposed authentication scheme achieves mutual authentication between the server and the client. The use of time-bound tickets reduces the server's processing overhead efficiently. The correspondence relationship between the digital ticket and the client's smart card prevents user masquerade attack effectively. By security analysis, we show that the proposed scheme is resistant to masquerade attack, replay attack and password guessing attack. By performance analysis, we show that the proposed scheme has good efficiency. The proposed scheme is very suitable for cloud computing.

Keywords: cloud computing, mutual authentication, digital ticket, masquerade attack.

1 Introduction

Cloud Computing [1, 2] has become a very popular technology. By using cloud computing, both the data storage and the computation resources are moved from personal computers into the cloud. Recently many companies provide the cloud storage services, including Amazon Simple

Storage Service (S3) [3], Microsoft SkyDrive [4], Nirvanix CloudNAS [5], etc. As people store their important data in the cloud without keeping a local copy, it is important for the cloud clients to be able to verify the data integrity and availability.

Remote data integrity checking protocols [6–10] are proposed to ensure the data integrity in the cloud. Ateniese et al. [6] propose the technology of provable data possession to enable the client to verify the data integrity without retrieving it from the server. Juels and Kaliski [7] propose the scheme called proofs of retrievability to enable the server to produce a concise proof about the data availability. Other schemes [8–10] have also been proposed with increased scalability and efficiency. However, as the clients of cloud services are numerous and are also increasing very quickly, these technologies bring a lot of extra computation and storage overhead to the server. Mechanisms should be designed to reduce these overhead as much as possible.

One possible solution is to limit the client's data verification frequency. In this method, the server releases a certain number of digital tickets to the client at a constant frequency, e.g., 12 tickets per year. The client uses one ticket for one time of data verification, and after that, the used ticket becomes invalid and cannot be reused. If the client needs more tickets, she must purchase them from the server. On the other hand, the client's identity should be properly authenticated to the server before she uses these services.

Recently Lin and Chang [11] propose a countable and time-bound password-based user authentication scheme, which is a possible method for solving this problem. In the Lin-Chang scheme, the tickets $\{TK_1, TK_2, ..., TK_t\}$ are generated by the server in a way that the authentication value in TK_j is one of the square roots of the value in TK_{j+1}. The Lin-Chang scheme is secure against masquerading attack and replay attack, and also achieves good efficiency at the client side. However, the Lin-Chang scheme has several disadvantages, which make it not suitable in this environment. Firstly, the ticket in Lin-Chang scheme is not associated with the client's identity, which allows anyone who obtains the ticket to be able to use it. Secondly, the client's tickets can only be used sequentially. Whenever TK_j expires, the tickets $TK_1, TK_2, ..., TK_{j-1}$ cannot be used anymore. Thirdly, the Lin-Chang scheme involves high-cost modular exponentiation operations at the server side, which bring large amount of overhead as the quantity of cloud clients increases quickly. Finally, the Lin-Chang scheme does not achieve mutual authentication.

In this paper, we propose a new ticket-based mutual authentication scheme which has improvements in these aspects. Firstly, we use smart card in the mutual authentication scheme. Each client has her own unique smart card. The client's tickets are associated with her smart card, so that even when her tickets are lost, they cannot be used by other clients. Secondly, in the proposed scheme, tickets are relatively independent of one another, so that the tickets need not be used sequentially. When one ticket expires, all other tickets that do not expire can still be used. Thirdly, the proposed scheme involves only lightweight exclusive-or and hash computations, which makes it very efficient at both the client side and the server side. Finally, the proposed scheme achieves mutual authentication, so that both the server and the client authenticates each other when performing the data verification. By security analysis, we show that the proposed scheme is secure against lost smart card attack, lost ticket attack, masquerade attack, replay attack, etc. By performance analysis, we show that the proposed scheme achieves good performance at both the server side and the client side. The proposed scheme is suitable for cloud computing.

The rest of this paper is organized as follows. Section 2 presents the system model and introduces common notations used throughout this paper. In section 3, the proposed time-bound ticket-based mutual authentication scheme is presented. In section 4, we show that the proposed scheme is secure against adversary's attacks. In section 5, we show that the proposed scheme is cost-efficient. Finally, we conclude in section 6.

2 System Model and Common Notations

In the proposed scheme, there are two different entities: the cloud server and the client. The cloud server provides data storage services to a lot of clients, and also provides data integrity verification services to the clients. Furthermore, the cloud server is in charge of the client registration, client authentication and digital ticket management. Clients put their data at the server and retrieves the data on demand. Each client has a unique identification and a password with which she can prove her identity to the server. In addition, similar to the Lin-Chang scheme [11], a bulletin board is maintained by the server for preventing repeated usage of one ticket.

For convenience of description, we denote the cloud server by S and the clients by $\{U_i, i = 1, 2, 3, ...\}$. Denote the identification and password of U_i by ID_i and PW_i. We use a cryptographic hash function $h()$ and a keyed hash function $h_K()$, with K as a cryptographic key. S has two long-term secret keys, denoted by K_1 and K_2.

3 The Proposed Mutual Authentication Scheme

In this section, we present the proposed time-bound ticket-based mutual authentication scheme. The proposed scheme consists of 4 phases: registration phase, verification request phase, mutual authentication phase and password change phase.

3.1 Registration Phase

When the client U_i initially registers herself to S, the registration phase is invoked. The registration phase consists of the following steps:

1. U_i selects her own ID_i, her password PW_i and a random number b. Then U_i computes $IPB_i = H(ID_i||H(PW_i \oplus b))$ and sends $\{ID_i, IPB_i, t\}$ to the server, in which t is the number of digital tickets U_i needs. At the same time U_i pays the corresponding ticket fee to S. The ticket fee can be payed in the form of either real or virtual currency, which depends on the policy of S.

2. After S receives the message and ticket fee from U_i, S generates t tickets for U_i. Denote the jth ticket of U_i by $T_i^{(j)}$. Denote by $TID_i^{(j)}$ and $VP_i^{(j)}$ the ticket ID and valid period of $T_i^{(j)}$ respectively. Specifically, S generates $\{(TID_i^{(j)}, VP_i^{(j)}),\ j = 1, 2, ..., t\}$ and computes as follows:

$$W_i = IPB_i \oplus H(ID_i,\ K_1),$$

$$\alpha_i^{(j)} = H_{K_2}(ID_i \parallel TID_i^{(j)} \parallel VP_i^{(j)}),$$

$$\beta_i^{(j)} = \alpha_i^{(j)} \oplus IPB_i.$$

$T_i^{(j)}$ has two parts:

$$T_i^{(j)} = (T_i^{(j)_1}, T_i^{(j)_2}),$$

in which

$$T_i^{(j)_1} = (TID_i^{(j)},\ VP_i^{(j)}),$$

$$T_i^{(j)_2} = \beta_i^{(j)}.$$

S also computes $Z_i = H_{K_2}(ID_i) \oplus IPB_i$, which is used during the password change phase.

3. S writes ID_i, t, W_i, Z_i and $T_i^{(j)},\ j = 1, 2, ..., t$ into a smart card and sends it to U_i.

4. When U_i receives her smart card, she writes b into the card.

3.2 Verification Request Phase

As the client receives t tickets, she can use these tickets to perform data verification at most t times. The description below assumes that this is the kth verification request that U_i invokes.

1. U_i inserts her smart card into a card reader and enters ID_i and PW_i.
2. The smart card generates a nonce r_U according to system time, and computes

$$IPB_i = H(ID_i||H(PW_i \oplus b)), H_i = W_i \oplus IPB_i \ ;$$

$$C_1 = r_U \oplus H_i, C_2 = H(r_U) \oplus T_i^{(k)_2} \oplus IPB_i.$$

Then U_i's smart card sends $\{ID_i, T_i^{(k)_1}, C_1, C_2\}$ to S.

3.3 Mutual Authentication Phase

When S receives the verification request message from U_i, it executes the following steps:

1. S checks the validity of ID_i and rejects the service request if ID_i is invalid.
2. S checks whether the ticket ID $TID_i^{(k)}$ is on the bulletin board. If it is on the bulletin board, then S rejects U_i's service request and terminates the process.
3. S checks whether the current date is in the range of $VP_i^{(k)}$ or not. If not, then S rejects U_i's service request and terminates the process.
4. S computes $D_0 = H(ID_i, K_1)$, $D_1 = C_1 \oplus D_0$ and $D_2 = H(D_1) \oplus C_2$.
5. S computes $H_{K_2}(ID_i \parallel TID_i^{(k)} \parallel VP_i^{(k)})$ and checks whether it is equal to D_2. If they are not equal, then S rejects U_i's service request and terminates the process. Otherwise, S authenticates U_i successfully.
6. S generates a random nonce r_S, computes $C_3 = D_0 \oplus r_S$ and $C_4 = H(r_U, r_S)$ and sends $\{C_3, C_4\}$ to U_i. S also computes $K_S = H(D_0, r_U||r_S)$, which will be used as a subsequent session key.
7. U_i's smart card computes $D_3 = C_3 \oplus H_i$. Then it compares $H(r_U, D_3)$ with C_4. If they are equal, U_i authenticates S successfully. Then the smart card computes $K_C = H(H_i, r_U||r_S)$ and uses K to communicate with S in the subsequent data verification process. Note that $K_C = H(H_i, r_U||r_S) = H(H(ID_i, K_1), r_U||r_S) = K_S$. When the kth data verification is finished, U_i deletes the ticket $T_i^{(k)}$ from its smart card, and S publishes $TID_i^{(k)}$ to its bulletin board. Finally the smart card deletes the nonce r_U, and S deletes r_S from its memory, to prevent replay attack.
8. After the mutual authentication, the data verification is performed. The data verification schemes [6–10] are independent of the proposed authentication scheme, so we don't describe the details. The keys K_C and K_S can be used for secret communication.

3.4 Password Change Phase

This phase is invoked when U_i needs to change her password PW_i to a new one. The password change phase consists of the following steps:

1. U_i inserts her smart card into a card reader, and enters ID_i and PW_i.
2. The smart card generates a nonce r_U according to system time, and computes

$$IPB_i = H(ID_i||H(PW_i \oplus b)),$$

$$C_1 = r_U \oplus W_i \oplus IPB_i, \ C_2 = H(r_U) \oplus Z_i \oplus IPB_i.$$

Then U_i's smart card sends $\{update, ID_i, C_1, C_2\}$ to S, in which *update* is the message type indicating this is a password change request message.

3. When S receives this message, it checks the validity of ID_i. If ID_i is invalid, it rejects the service request.

4. S computes $D_1 = C_1 \oplus H(ID_i, K_1)$ and $D_2 = H(D_1) \oplus C_2$. After that S checks whether D_2 is equal to $H_{K_2}(ID_i)$. If they are not equal, S rejects the password change request. Otherwise, S authenticates U_i successfully and accepts the password change request.

5. S generates a random nonce r_S, computes $C_3 = H(ID_i, K_1) \oplus r_S$ and $C_4 = H(r_U, r_S)$ and sends $\{C_3, C_4\}$ to U_i.

6. The smart card computes $D_3 = C_3 \oplus W_i \oplus IPB_i$. Then it compares $H(r_U, D_3)$ with C_4. If they are equal, the smart card successfully authenticates S and prompts U_i to enter a new password.

7. U_i enters a new password PW_i^{new}. Then the smart card computes $IPB_i^{new} = H(ID_i || H(PW_i^{new} \oplus b))$. After that the smart card computes $W_i^{new} = W_i \oplus IPB_i \oplus IPB_i^{new}$, which yields $H(ID_i,\ K_1) \oplus IPB_i^{new}$, and computes $Z_i^{new} = Z_i \oplus IPB_i \oplus IPB_i^{new}$, which yields $H_{K_2}(ID_i) \oplus IPB_i^{new}$. The smart card also updates $T_i^{(j)_2}$ to $T_i^{(j)_2} \oplus IPB_i \oplus IPB_i^{new}$ for all remaining tickets, which yields $\alpha_i^{(j)} \oplus IPB_i^{new}$.

4 Security Analysis of the Proposed Scheme

In this section, we present security analysis of the proposed scheme. In section 4.1, we show that the server's secret key cannot be obtained by an adversary who monitors the communication. In sections 4.2-4.5, we show that the proposed scheme is resistant to lost smart card attack, masquerade attack, replay attack and valid period extending attack.

4.1 Secrecy of the server's secret key

In the proposed scheme, communications between U_i and S are through a common channel. So we assume the adversary can eavesdrop this channel and get any messages transmitted between U_i and S.

During the kth verification request and mutual authentication phase, the adversary can get messages $\{ID_i, T_i^{(k)_1},\ C_1, C_2, C_3, C_4\}$, in which ID_i and $T_i^{(k)_1}$ have no relationship with K_1 or K_2, and C_1, C_2, C_3, C_4 are as follows:

$$C_1 = r_U \oplus W_i \oplus IPB_i = r_U \oplus H(ID_i,\ K_1)$$

$$C_2 = H(r_U) \oplus T_i^{(k)_2} \oplus IPB_i = H(r_U) \oplus H_{K_2}(ID_i \parallel TID_i^{(k)} \parallel VP_i^{(k)})$$

$$C_3 = H(ID_i, K_1) \oplus r_S$$

$$C_4 = H(r_U, r_S)$$

As r_U and r_S are both random nonces generated by U_i and S, the adversary cannot guess their values. In addition, due to the one-wayness of hash function, the adversary cannot get r_U or r_S from $H(r_U, r_S)$. So the adversary cannot get the value of $H(ID_i,\ K_1)$ or $H_{K_2}(ID_i \parallel TID_i^{(k)} \parallel VP_i^{(k)})$. As a result, the adversary cannot get any information on S's secret keys by eavesdropping channels.

4.2 Resistance to Attacks Based on Lost Smart Card

A user's smart card may get lost due to an accident or the user's carelessness. In this section, we show that when an adversary gets a lost smart card, he cannot carry out attacks to the proposed scheme. We first show that the proposed scheme is resistant to offline password guessing attack. After that, we show that the lost tickets cannot be used by the adversary.

Resistance to Offline Password Guessing Attack

When an adversary gets the lost smart card of U_i, he can extract the stored data from it by monitoring the power consumption [12] or analyzing the leaked information [13]. The values stored in U_i's smart card are

$$ID_i,\ t,\ W_i,\ Z_i,\ b,\ T_i^{(j)},\ j = 1, 2, ..., t.$$

The adversary can pick up a password candidate PW'. Then he can computes $IPB_i' = H(ID_i||H(PW_i' \oplus b))$. From $W_i = IPB_i \oplus H(ID_i,\ K_1)$, $Z_i = H_{K_2}(ID_i) \oplus IPB_i$ and $\beta_i^{(j)} = \alpha_i^{(j)} \oplus IPB_i = H_{K_2}(ID_i \parallel TID_i^{(j)} \parallel VP_i^{(j)}) \oplus IPB_i$ he can further compute $H'(ID_i,\ K_1) = W_i \oplus IPB_i'$, $H'_{K_2}(ID_i) = Z_i \oplus IPB_i'$ and $H'_{K_2}(ID_i \parallel TID_i^{(j)} \parallel VP_i^{(j)}) = \beta_i^{(j)} \oplus IPB_i'$. However, as K_1 and K_2 are S's secret keys, the adversary cannot get them. So the adversary cannot test whether PW' is equal to PW_i from these values.

From the above analysis, we can see that the proposed scheme is resistant to offline password guessing attack.

Resistance to Attacks Based on Lost Tickets

When an adversary gets the lost smart card of U_i, he can extract the stored tickets $T_i^{(j)},\ j = 1, 2, ..., t$ from it. However, as the adversary cannot get PW_i from offline guessing attack (refer to 4.2), he cannot compute $IPB_i = H(ID_i||H(PW_i \oplus b))$. As a result, the adversary cannot make a valid verification request message $\{C_1, C_2\}$, because both the computations of C_1 and C_2 require the knowledge of IPB_i.

From the above analysis, we can see that the proposed scheme is secure when an adversary gets a lost ticket.

4.3 Resistance to Masquerade Attack

Suppose the adversary gets a set of history messages during the past channel eavesdropping. In order to masquerade as U_i, the adversary has to forge a valid message $\{ID_i^*, T_i^{(k)_1^*}, C_1^*, C_2^*\}$ which can pass S's authentication process.

It's easy to see that the computation of $C_1^* = r_U \oplus H(ID_i,\ K_1)$ requires the adversary to get knowledge of $H(ID_i,\ K_1)$. From 4.1 we know that the adversary cannot get $H(ID_i,\ K_1)$, so he cannot forge a valid verification request message either.

On the other hand, if the adversary wants to masquerade as the server, he must be able to compute a valid message $\{C_3, C_4\}$, in which $C_3 = H(ID_i, K_1) \oplus r_S$ and $C_4 = H(r_U, r_S)$. Because the computation of C_3 requires the knowledge of $H(ID_i, K_1)$, which the adversary does not have, the adversary cannot masquerade as the server.

From the above analysis, we can see that the proposed scheme is resistant to masquerade attack.

4.4 Resistance to Replay Attack

In the proposed scheme, the unique ticket ID and the nonces are used for preventing replay attack.

Assume that the adversary gets a valid message $\{ID_i, T_i^{(k)_1}, C_1, C_2\}$ by eavesdropping communications between U_i and S. Then he tries to resend the message to S after a period of time when the mutual authentication process finishes, with the expectation of obtaining data verification service. However, when S receives this message, it will find that the ticket ID $TID_i^{(k)}$ is

already on the bulletin board. So S will reject the adversary's service request and terminate the process.

On the other hand, if the adversary gets a message from S to U_i: $\{C_3 = H(ID_i, K_1) \oplus r_S, C_4 = H(r_U, r_S)\}$. The adversary may try to resend it to U_i after a period of time when the mutual authentication process is finished. However, when U_i's smart card receives $\{C_3, C_4\}$, the nonce r_U has already been deleted from its storage. So this message will be discarded immediately.

From the above analysis, we can see that the proposed scheme is resistant to replay attack.

4.5 Resistance to Valid Period Extending Attack

Whenever U_i wants to use a ticket whose valid period does not include the current date, the protocol can ensure that the ticket cannot be used even if U_i tries to modify the ticket's begin date or expiration date.

Assume U_i wants to use ticket $T_i^{(k)}$ by changing the ticket's $VP_i^{(k)}$ to $\hat{VP}_i^{(k)}$, so that the current date is included in $\hat{VP}_i^{(k)}$. Denote the modified ticket by $\hat{T}_i^{(k)}$. When U_i sends the message $\{ID_i, \hat{T}_i^{(k)1}, C_1, C_2\}$ to S, S computes $D_1 = C_1 \oplus H(ID_i, K_1)$ and $D_2 = H(D_1) \oplus C_2$. Then S computes $H_{K_2}(ID_i \parallel TID_i^{(k)} \parallel \hat{VP}_i^{(k)})$ and compares it with D_2. Since $D_2 = H(D_1) \oplus C_2 = H(C_1 \oplus H(ID_i, K_1)) \oplus C_2 = \alpha_i^{(k)} = H_{K_2}(ID_i \parallel TID_i^{(k)} \parallel VP_i^{(k)}) \neq H_{K_2}(ID_i \parallel TID_i^{(k)} \parallel \hat{VP}_i^{(k)})$, S will reject U_i's verification request and terminate the process immediately.

From the above analysis, we can see that the server can limit the data verification frequency by specifying a valid period for each ticket. For example, the typical coverage of the valid period can be one week, one month, one year, etc. The proposed scheme can prevent the client from using a ticket whose valid period does not include the current date.

5 Performance Analysis

In this section we present performance analysis of the proposed scheme. The main operations include the hash computation and the exclusive-or operations, which are summarized in Table 1. We use the hash-based message authentication code (HMAC) [14] as the keyed hash, which incurs 2 exclusive-or and 2 common hash operations.

Table 1: Performance Analysis of the Proposed Scheme

	operation	verification request phase	mutual authentication phase
client	*xor*	5	1
	hash	3	2
	keyed hash	0	0
server	*xor*	0	3
	hash	0	4
	keyed hash	0	1

Compared with [11], which uses expensive operations like modular exponentiations, our scheme is much more efficient. According to the Crypto++ benchmarks[1] [15], the SHA-256 hash algorithm [16] can achieve a throughput of 111MiB/Sec under Intel Core2 1.83GHz processor. In the proposed scheme, the length of a message to be hashed does not exceed 1KB, so the average time for one hash computation is about 0.009ms. If the Lin-Chang scheme uses

[1]The Crypto++ 5.6.0 benchmark is evaluated in Intel Core2 1.83 GHz processor under Windows Vista in 32-bit mode

the primes of length 1024bits, then one time of modular exponentiation will cost about 1.46ms under the same platform [15]. Our scheme is at least 40 times more efficient than the Lin-Chang scheme [11] in case of the server's computation time. This is a tremendous performance improvement to the cloud servers.

Performance Benefit by Limiting Verification Frequency The cloud server can also benefit its performance by limiting the client's data verification frequency. This is achieved by controlling the valid period of the digital ticket. From the performance evaluation of [6] we know that for a 30MB file, the computation time of the data integrity verification is approximately 1 second[2] at the server side. Assume the cloud storage system has 100,000 clients, and each client performs data integrity verification once a week. Then the average computation time of the cloud server during one day is approxiately:

$$\frac{100,000}{7} \cdot 1s \approx 3.97 \; hours.$$

By using the proposed scheme, the server can limit the data verification frequency to once per month, or even once per quarter, so that the average computation time of the cloud server is reduced to 56 minutes per day and 18.5 minutes per day respectively.

6 Conclusions

In this paper, we propose a time-bound ticket-based mutual authentication scheme. In the proposed scheme, the digital tickets are associated with the client's smart card, which effectively prevents the ticket from being used by other clients. By designing a mutual authentication based on the client's smart card, both the server and the client are assured of each other's identity. The proposed authentication scheme can efficiently decrease the server's processing overhead by limiting the data verification frequency. By security analysis and performance analysis, the proposed scheme is shown to be both secure and efficient. It is very suitable for providing mutual authentication in cloud computing.

Acknowledgment

This work was supported by NSF CNS-0845149, NSF CCF-0915374 and Knowledge Innovation Program of Chinese Academy of Sciences (No. YYYJ-1013).

Bibliography

[1] B. Hayes, "Cloud computing," *Commun. ACM*, vol. 51, no. 7, pp. 9–11, 2008.

[2] C. Cachin, I. Keidar, and A. Shraer, "Trusting the cloud," *SIGACT News*, vol. 40, no. 2, pp. 81–86, 2009.

[3] Amazon.com, "Amazon Web Services (AWS)," *http://aws.amazon.com/s3/*, 2009.

[4] Microsoft.com, "Microsoft Windows SkyDrive," *http://windowslive.com/online/skydrive*, 2009.

[2]The experiment environment in [6] is Intel 2.8 GHz Pentium IV system with a 512 KB cache, an 800 MHz EPCI bus, and 1024 MB of RAM.

[5] Nirvanix.com, "Nirvanix cloudNAS," *http://www.nirvanix.com/products-services/*, 2009.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 598–609, ACM, 2007.

[7] A. Juels and B. S. Kaliski, Jr., "Pors: proofs of retrievability for large files," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 584–597, ACM, 2007.

[8] E.-C. Chang and J. Xu, "Remote integrity check with dishonest storage server," in *13th ESORICS*, pp. 223–237, Springer Berlin / Heidelberg, 2008.

[9] A. Heitzmann, B. Palazzi, C. Papamanthou, and R. Tamassia, "Efficient integrity checking of untrusted network storage," in *StorageSS '08*, pp. 43–54, ACM, 2008.

[10] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: a high-availability and integrity layer for cloud storage," in *CCS '09*, (New York, NY, USA), pp. 187–198, ACM, 2009.

[11] I.-C. Lin and C.-C. Chang, "A countable and time-bound password-based user authentication scheme for the applications of electronic commerce," *Information Sciences*, vol. 179, no. 9, pp. 1269 – 1277, 2009.

[12] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, (London, UK), pp. 388–397, Springer-Verlag, 1999.

[13] T. Messerges, E. Dabbish, and R. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541–552, 2002.

[14] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," *RFC2104*, February 1997.

[15] "Crypto++ 5.6.0 benchmarks," *http://www.cryptopp.com/benchmarks.html*.

[16] "Secure hash standard," *Federal Information Processing Standards Publication 180-2*, August 2002.