Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 19(11): 11034–11046.
DOI: 10.3934/mbe.2022514
Received: 27 June 2021
Revised: 10 August 2021
Accepted: 27 May 2022
Published: 02 August 2022

Research article

Enabling secure mutual authentication and storage checking in cloud-assisted IoT

Dengzhi Liu[1,2], Zhimin Li[1], Chen Wang[3,*], and Yongjun Ren[3]

[1] School of Computer Engineering, Jiangsu Ocean University, Lianyungang 222005, China
[2] Jiangsu Institute of Marine Resources Development, Lianyungang 222005, China
[3] School of Computer Science, Nanjing University of Information Science and Technology, Nanjing 210044, China

* **Correspondence:** Email: wangchennuist@126.com.

Abstract: Internet of things (IoT) is a technology that can collect the data sensed by the devices for the further real-time services. Using the technique of cloud computing to assist IoT devices in data storing can eliminate the disadvantage of the constrained local storage and computing capability. However, the complex network environment makes cloud servers vulnerable to attacks, and adversaries pretend to be legal IoT clients trying to access the cloud server. Hence, it is necessary to provide a mechanism of mutual authentication for the cloud system to enhance the storage security. In this paper, a secure mutual authentication is proposed for cloud-assisted IoT. Note that the technique of chameleon hash signature is used to construct the authentication. Moreover, the proposed scheme can provide storage checking with the assist of a fully-trusted entity, which highly improves the checking fairness and efficiency. Security analysis proves that the proposed scheme in this paper is correct. Performance analysis demonstrates that the proposed scheme can be performed with high efficiency.

Keywords: Internet of things; cloud computing; mutual authentication; storage checking

1. Introduction

Internet of Things is a promising technology that may promote human society towards intelligence. However, the constrained local storage space of the devices will be an obstacle for the further development of IoT. Cloud computing is a new emerging technology in recent years. By accessing the wired or wireless network, clients can enjoy the high-quality cloud services. At present, the resources of storage and computing are the main services that cloud computing can provide for consumers [1–3]. The resource of clients is constrained due to the IoT devices in the system are lightweight. Therefore, the resource-constrained clients of IoT can outsource their local data to cloud servers for long-term data

storage. Moreover, clients can enjoy the unlimited storage resource and computing capability anytime and anywhere via the Internet [4, 5]. As the cloud server can provide virtual resources for clients, they no longer need to worry about the local hardware investments [6]. Clients only need to pay the fees of cloud usage as a manner of pay-as-you-use. Hence, with the assist of cloud computing, the technology of IoT may have the further development space.

In the practical cloud usage, the adversaries may impersonate legitimate clients to cheat cloud server, which brings a great security threats to the data stored in cloud servers [7, 8]. On the other hand, cloud servers can be accessed by any users through the Internet, which is easy to cause malicious users to attack the server [9, 10]. To ensure the storage security and avoid server attack, the authentication protocol should be provided to ensure the security of the system [11–15]. In 2009, Li et al. proposed an identity-based authentication protocol based on identity-based encryption and signature schemes [16]. Due to the superior feature of identity-based cryptography, The scheme of Li et al. is more efficient and very suitable for lightweight clients. To realize more strong security, Choudhury et al. proposed a strong user authentication framework for cloud computing that can strongly verify the legitimacy of users before the participation in the system [17]. Moreover, The scheme of Choudhury et al. can provide the properties of identity management, mutual authentication, session key establishment between cloud users and servers. In 2015, Liu et al. proposed a privacy-preserving authentication protocol based on shared authority [18]. The scheme of Liu et al. can also provide data sharing based on proxy re-encryption. In addition, various security features can realize except authentication in the scheme of Liu et al., such as data anonymity, user privacy, and forward security.

However, the previous schemes only focused on the research of authentication security in cloud computing but ignored the storage security. Moreover, there is no in-depth study on the issue of mutual authentication. Despite some researches have devoted themselves to the research of mutual authentication in this respect, the traditional interactive authentication protocol will bring much communication cost to clients and servers. Hence, a secure mutual authentication with less interacting should be designed. In addition, the storage checking should also be considered in the design of the authentication scheme in cloud-assisted IoT.

1.1. Contributions

In this paper, a secure mutual authentication and storage checking scheme is proposed for cloud-assisted IoT. The main contributions of the proposed scheme are shown as follows.

(1) Chameleon hash signature is used to design the authentication mechanism, which improves the security of authentication. Moreover, the proposed scheme is extended to support mutual authentication between cloud servers and clients that ensures the storage security for clients, and avoid the participation of malicious entities in the system.
(2) By using the encryption of certificate in the authentication of server, the proposed scheme effectively prevents the leakage of sensitive information. Moreover, the proposed scheme can resist the tamper attack and the replay attack.
(3) The proposed scheme also supports storage checking for the stored data in cloud servers. Moreover, the storage checking tasks are delegated to the entity of Authority, which improves the fairness of the storage checking and highly reduces the computational overhead at the client side.

1.2. Related works

The authentication protocol can ensure the security of communication and storage security in cloud computing [19–22]. To enhance the security of mobile users in cloud computing, Tsai et al. proposed an efficient authentication for distributed mobile cloud computing [23]. By using the bilinear pairing and dynamic nonce generation, the proposed scheme in [23] can provide mutual authentication, key exchange, user anonymity and so on. In the scheme of Tsai et al., only the target server needs to interact with the request client, which highly reduces the communication cost between the client and the server. To protect the access policies in cloud computing, Liu et al. proposed a hierarchical attribute based access control scheme that can achieve scalability and authentication in realizing write privilege on the stored data [24]. Note that the scheme of Liu et al. can well enhance the security of access control on stored data. To solve the authentication of IoT devices in cloud computing, Amin et al. proposed an architecture that can implement authentication protocol using smartcard [25]. In addition, AVISPA tool and BAN logic model are used in the design of the proposed scheme in the scheme of Amin et al., which ensures the legitimate users can access the sensitive information from the private cloud with various security threats resistance.

1.3. Organization

The rest of this paper is organized as follows. Section 2 presents the preliminaries. Section 3 provides the system model and the design goals. Section 4 introduces the proposed scheme in detail. Section 5 describes the analysis of the security and the performance of the proposed scheme. The conclusion of this paper is given in Section 6.

2. Preliminaries

In this section, the preliminaries of this paper are provided, including bilinear pairing and chameleon hash signature. The detailed introduction of them is shown as follows.

2.1. Bilinear pairing

Bilinear pairing is proposed by Boneh et al. in 2001 and it is widely used to construct all kinds of cryptographic security protocols [26]. In general, bilinear pairing are constructed on elliptic curve or hyperelliptic curve in finite field. In the construction of bilinear pairing, two multiplicative cyclic groups $\mathbb{G}_0$ and $\mathbb{G}_1$ with large prime order q are defined. The bilinear pairing can be denoted as $\hat{e}$: $\mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. Then, two points $\mathcal{P}, \mathcal{Q} \in \mathbb{G}_0$ and two integers $a, b \in \mathbb{Z}_q^*$ are randomly selected. Here, $\mathbb{Z}_q^*$ is a set that consists integers 1 to q-1 with modulo q. The generator of group G_0 is $\mathcal{G}$. The properties of the bilinear pairing are shown as follows.

(1) Bilinear: $\hat{e}\left(\mathcal{P}^a, \mathcal{Q}^b\right) = \hat{e}\left(\mathcal{P}, \mathcal{Q}\right)^{ab}$;
(2) Non-degenerate: $\hat{e}\left(\mathcal{G}, \mathcal{G}\right) \neq 1$;
(3) Computable: $\hat{e}\left(\mathcal{P}, \mathcal{Q}\right)$ can be computed by an existing algorithm.

2.2. Chameleon hash signature

Chameleon hash signature evolved from the technique of chameleon hashing that is a non-interactive signature algorithm [27]. That is to say, the prover generates the signature without in-

teracting with the verifier in chameleon hash signature. The main process of chameleon hash signature is shown as follows.

First, the prover selects $s \in_R \mathbb{Z}_q^*$ as its secret key. Then, the prover computes the chameleon as $C = \mathcal{G}^s$. To realize the authentication, the prover selects private key $a \in_R \mathbb{Z}_q^*$ and computes the corresponding public key as $P = \mathcal{G}^a$. After that, the prover generates an auxiliary parameter $m_{au} = CFind(a, nonce, s) = s - a\lambda$, where $CFind$ is a collision finding algorithm, $nonce$ is authentication challenge and $\lambda = H(P \oplus nonce)$. The information of $\{C, m_{au}, P, nonce\}$ will be sent to the prover for the further authentication. Upon receiving the authentication information from the prover, the verifier will compute a chameleon hash parameter as $CH(m_{au}, P, nonce) = \mathcal{G}^{m_{au}} \cdot P^{\lambda}$. According to checking whether the chameleon hash parameter is equal to C, the verifier can successfully authenticate the prover.

3. Problem statement

This section provides the system model of the proposed scheme. Moreover, the design goals of the proposed scheme are given, including mutual authentication, tamper-resistance and storage checking.

3.1. The system model

The system model of the proposed scheme consists of three different kinds of entities, which are Clients, Authority and Cloud Servers. The system model of the proposed scheme is shown as Figure 1. The detailed introduction and security features of each entity are shown as follows.

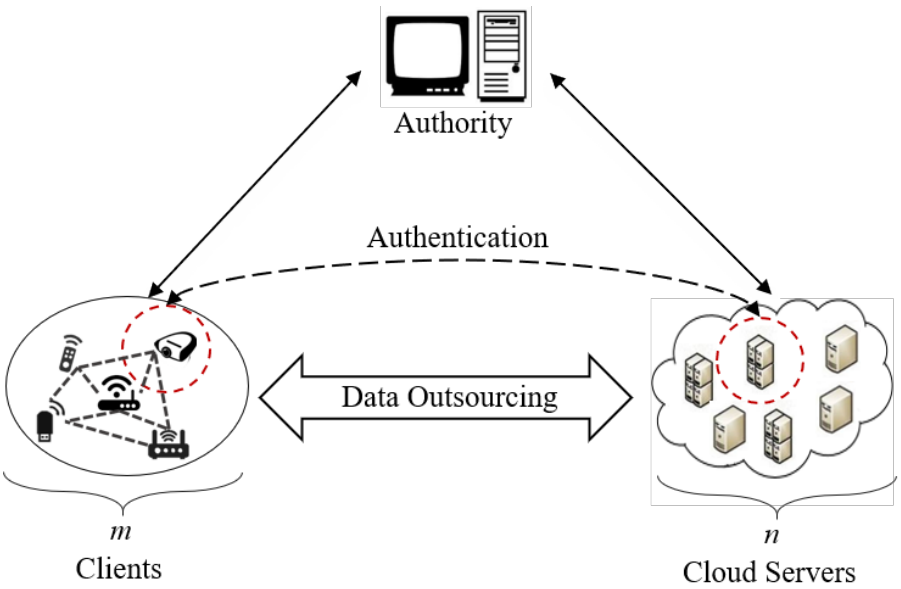

Figure 1. The system model.

(1) **Clients**. In the system, clients are the IoT devices who enjoy the cloud services as a manner of pay-per-use through the Internet [28]. The IoT clients are not fully-trusted in the system. Note that the storage resource of clients is limited. To realize the unconstrained data file storage,

clients can outsource the data files to cloud servers for long-term storage [29]. However, clients cannot determine whether the servers are legal. Hence, clients should authenticate the validity of a specific server before the data outsourcing. Moreover, clients can check the data file storage in the cloud servers with the help of Authority. Here, clients are honest, but it is also easy for attackers to impersonate legitimate users to participate in the system.

(2) **Authority**. Compared to clients and cloud servers, the entity of Authority is a fully-trusted entity. Authority is responsible for generating the required security parameters and certificates for clients and cloud servers in the system. Moreover, Authority can check the data file storage on behalf of clients.

(3) **Cloud Servers**. The cloud servers are distributed and can be connected each other by the Internet. However, cloud servers are honest-but-curious to the stored data. Moreover, cloud servers are vulnerable to be attacked by adversaries. In addition, cloud servers may destroy or delete the infrequently used data file for financial reasons. Hence, cloud servers need to authenticate the validity of the accessed clients and cooperate with the user's data storage checking at any time during the service providing period.

3.2. Design goals

(1) **Mutual Authentication**. To ensure the storage security and avoid the participation of illegal entities in the system, the proposed scheme should provide a mechanism of mutual authentication. That is to say, every client can authenticate the validity of any one of the cloud servers in the system and each cloud server can implement the authentication of any one of the clients.

(2) **Tamper-Resistance**. In order to avoid the influence of external factors on the authentication, the proposed scheme should provide the property of tamper-resistance for the authentication. In other words, any attacker cannot modify the authentication information of the entity that needs to be authenticated in the system.

(3) **Storage Checking**. The proposed scheme should support the storage checking. To improve the fairness and alleviate the computational overhead of the storage checking, the clients in the system can delegate the storage checking tasks to the entity of Authority.

4. The proposed scheme

The proposed scheme consists of two parts, including authentication and storage checking. The authentication can realize mutual authentication, namely, the mutual authentication between clients and cloud servers. With the assist of mutual authentication, the proposed scheme can avoid illegal and attacked entities from participating in the system. The mutual authentication is designed based on the idea of chameleon hashing in [30]. Moreover, the proposed scheme supports storage checking, which highly improves the security of the data file storage. The detailed description of the proposed scheme is shown as follows.

4.1. Authentication construction

In the system, the total number of the clients and cloud servers is m and n, respectively. Assume that client x is a data owner who wants to enjoy the cloud services provided by server r via the Internet, where $1 \leqslant x \leqslant m$ and $1 \leqslant r \leqslant n$. However, client x cannot determine whether server r is an authorized

server by the Authority. To avoid the harm of malicious servers to users' rights and interests, in the proposed scheme, the clients in the system can authenticate the server before data outsourcing. At the same time, the cloud servers can also authenticate the validity of the accessed clients. The detailed construction of the proposed scheme is shown as follows.

First, client x selects $k_x \in_R \mathbb{Z}_p^*$ as its secret key. Then, client x computes a chameleon parameter as $CHA_x = \mathcal{G}^{k_x}$ by using for the secret key for the certificate computation, where $\mathcal{G}$ is a randomly selected point from group $\mathbb{G}_0$. After that, client x sends CHA_x to Authority in a secure channel. Upon receiving the chameleon parameter of client x, Authority will compute the certificate for client x as $C_x = S_{K_A}(CHA_x)$, where K_a is the secret key of Authority that is randomly selected from $\mathbb{Z}_p^*$. Note that S is an encryption algorithm and the corresponding public decryption key is pk_A. Similar to the certificate computation of client x, Authority can compute a certificate for server r according to the received chameleon parameter of server r. The certificate of server r is denoted as $C_r = S_{K_A}(CHA_r)$, where $CHA_r = \mathcal{G}^k$ and k_r is the secret key of server r.

In the authentication phase, client x randomly selects a private secret key sk_x from $\mathbb{Z}_p^*$. The corresponding public key can be computed as $pk_x = \mathcal{G}^{sk_x}$. Then, client x computes a security parameter as $SP_x = CFind(sk_x, nonce, k_x)$ for the further authentication. Finally, client x sends SP_x and its own certificate C_x to server r for verification. According to the received parameters from client x, server r can authenticate client x by checking whether $Verify = (C_x, pk_A) = CH(SP_x, pk_x, nonce)$ can hold using the public keys of Authority and client x. If the authentication of client x can pass, server r will compute encryption key $K_{xr} = sk_x \cdot pk_r$ by using its secret key and the public key of client x. Finally, server r encrypts its certificate C_r using K_{rx} as $E_{C_r} = Enc_{K_{rx}}(C_r)$. Here, the encryption of Enc is a symmetric encryption. The encrypted certificate of server r will be sent to client x for the validity checking. When receiving the encrypted certificate of server r, client x can compute the decryption key as $K_{rx} = sk_r \cdot pk_x$. Note that $K_{rx} = K_{xr}$. Hence, server r's encrypted certificate E_{C_r} can be decrypted by client x as $C_r^{'} = Dec_{K_{xr}}(E_{C_r})$ by using K_{xr}, where Dec is the decryption algorithm. According to checking whether C_r^* is equal to $C_r^{'}$, client x can determine whether server r is a honest server. Here, C_r^* is the certificate of server r stored in the certificate list at Authority side. If $C_r^* \neq C_r^{'}$, client x can check the correctness of equation $Verify(C_r, pk_A) = CH(SP_r, pk_r, nonce)$.

4.2. Storage checking details

When the mutual authentication between the client and the server is finished, the corresponding client can outsource its own data files to the valid cloud server. The main philosophy of storage checking in this paper is constructed based on the key techniques in studies [31, 32]. Suppose that the data file of client x is $F = [f_1, f_2, \cdots, f_z]$ and the data file identifier is Id_F. The tag of the data file can be computed as $t_F = Id_F \| SSig_{sk_s}(Id_F)$, where $SSig$ is a signature algorithm for tag generation and sk_s is the secret key of signature generation. The corresponding public key for signature verification is pk_s. After that, client x computes authenticator $\delta_i = \left(H_1(Id_F) \cdot \eta^{f_i}\right)^{k_x}$ for every data file block, where η is security parameter that is chosen from group $\mathbb{G}_0$ and $1 \leqslant i \leqslant z$. Moreover, client x computes $\omega = \hat{e}(\eta, CHA_x)$. Finally, client x uploads the local data file blocks with the authenticators and data file tag to cloud server r.

When client x wants to check the data file storage in server r, a specific data file storage checking request with security parameter ω needs to be generated and sent to Authority. Then, Authority retrieves the corresponding data file tag to local side and checks the correctness of it using pk_s. If the

tag can pass the checking, the entity of Authority recovers the data file identifier from t_F. To realize secure storage checking, Authority selects ρ parameters $v_i (1 \leqslant i \leqslant \rho)$ and randomly selects a security parameter c_i from $\mathbb{Z}_p$ for every v_i. Finally, Authority sends $\{i, c_i\}$ to server r as storage checking challenge.

Upon receiving the challenge from Authority, the cloud server r computes $fp_{rx} = \sum_{i\in[1,\rho]} (c_i \cdot f_i)$ and $tp_{rx} = \prod_{i\in[1,\rho]} \delta_i^{c_i}$ according to the stored data file and the corresponding tag. Then, server r sends fp_{rx} and tp_{rx} to Authority for the storage checking. The entity of Authority computes $\mu_f = \prod_{i\in[1,\rho]} \hat{e}\left((H(Id_F))^{c_i}, CHA_x\right)$ and checking whether $\mu_f \cdot \omega^{fp_{rx}} = \hat{e}(tp_{rx}, \mathcal{G})$ can hold.

5. Evaluation

In this section, the security and the performance of the proposed scheme are evaluated. The detailed evaluation is shown as follows.

5.1. Security analysis

Theorem 1. *The proposed authentication can be proved to be correct if all required parameters are generated successfully.*

Proof of Theorem 1. In the authentication of client x, we can get the chameleon hash result of SP_x and pk_x is $CH(SP_x, pk_x) = \mathcal{G}^{SP_x} \cdot pk_x^{\lambda}$. From the description of the authentication, the security parameter of SP_x can be computed as $SP_x = CFind(sk_x, k_x) = k_x - sk_x \cdot \lambda$. We have $CH(SP_x, pk_x) = \mathcal{G}^{k_x - sk_x\cdot\lambda} \cdot (\mathcal{G}^{sk_x})^{\lambda} = \mathcal{G}^{k_x}$. According to the computation of the certificate of client x, the certificate can be recovered by pk_A as $CHA_x = \mathcal{G}^{k_x}$. In addition, similar to the proof of client x authentication, the correctness of server r authentication can also be proved. Here, the detailed description of the server r authentication is omitted. Hence, the mutual authentication presented in the proposed scheme is correct. □

Theorem 2. *The encrypted certificates of servers can be successfully decrypted by using the generated keys of clients.*

Proof of Theorem 2. If $K_{xr} = K_{rx}$ is correct, the encrypted certificate of server r can be decrypted by client x. According to the above description, we can get that $K_{xr} = sk_x \cdot pk_r = sk_x \cdot (sk_r \cdot \mathcal{G}) = (sk_x \cdot \mathcal{G}) \cdot (sk_r) = pk_x \cdot sk_r$. The decryption of $K_{rx} = sk_r \cdot pk_x$. Therefore, the certificate decryption in the proposed scheme is correct. □

Theorem 3. *The correctness of the proposed storage checking mechanism can be proved .*

Proof of Theorem 3. If the storage checking equation can be proved to be correct, the proposed storage

checking in this paper is correct. The storage checking equation is solved as follows.

$$
\begin{aligned}
&\mu_f \cdot \omega^{fp_{rx}} \\
&= \prod_{i\in[1,\rho]} \hat{e}\left(\left(H\left(Id_F\right)\right)^{c_i}, CHA_x\right) \cdot \hat{e}\left(\eta, CHA_x\right)^{\sum_{i\in[1,\rho]}\left(c_i \cdot f_i\right)} \\
&= \prod_{i\in[1,\rho]} \hat{e}\left(\left(H\left(Id_F\right)\right)^{c_i}, \mathcal{G}^{k_x}\right) \cdot \prod_{i\in[1,\rho]} \hat{e}\left(\eta, \mathcal{G}^{k_x}\right)^{c_i \cdot f_i} \\
&= \prod_{i\in[1,\rho]} \hat{e}\left(\left(\left(H\left(Id_F\right) \cdot \eta^{f_i}\right)^{k_x}\right)^{c_i}, \mathcal{G}\right) \\
&= \hat{e}\left(\prod_{i\in[1,\rho]} \delta_i^{c_i}, \mathcal{G}\right) \\
&= \hat{e}\left(tp_{rx}, \mathcal{G}\right)
\end{aligned}
$$

That is to say, the proposed storage checking in this paper is correct.

□

5.2. Performance analysis

In order to present the efficiency of the proposed scheme, the computational cost at the different entity side in the system is analyzed. The symbols of $T_{E.}, T_{H.}, T_{Mul.}, T_{Add.}$ and $T_{P.}$ are used to denote the time required to execute the operations of exponentiation, hash function, multiplication, addition and pairing, respectively. The computational at each entity side is shown in Table 1. Note that the symbols of ρ and z is the number of the challenged data blocks and the total number of data blocks in the system, respectively. From Table 1, we can find that the computational cost of the client side is determined by the number of the total data blocks. The computational cost of the server side and the Authority side will be increased with the growth of the challenged data blocks. Table 2 shows the computational cost of authentication and storage checking in the proposed scheme. From Table 2, we can see that the computational cost of the phase of Authentication is almost a constant, but the computational cost of the server side will be larger with the increase of the total data blocks and the challenged data blocks.

Table 1. Computational cost at the different entity side.

Entity Side	**Computational Cost**
Client	$(2z+4)T_E + (3+z)T_{Mul.} + zT_{H.} + 1T_{Add.} + T_{P.}$
Authority	$\rho(2T_{Mul.} + T_{E.} + T_{Add.})$
Server	$(4+\rho)T_{E.} + (2\rho+3)T_{Mul.} + \rho T_{H.} + T_{P.}$

* $T_{E.}, T_{H.}, T_{Mul.}, T_{Add.}, T_{P.}$: Time required to perform the corresponding operations.
* ρ: The number of the challenged data blocks.
* z: The total number of data blocks.

Table 2. Computational cost of authentication and storage checking.

Phase	**Computational Cost**
Authentication	$7T_{E} + 5T_{Mul.} + 1T_{Add.}$
Storage Checking	$(2z + 2\rho + 1)T_{E.} + (z + 3\rho + 1)T_{Mul.} + (z + \rho)T_{H.} + \rho T_{Add.} + 2T_{P.}$

* $T_{E.}, T_{H.}, T_{Mul.}, T_{Add.}, T_{P.}$: Time required to perform the corresponding operations.
* ρ: The number of the challenged data blocks.
* z: The total number of data blocks.

In Table 3, the communication cost in the phase of authentication and storage checking is given. Here, we use the symbol of $l_{\mathbb{G}}$ and $l_{\mathbb{Z}}$ denotes the bit-length of parameters in group and integer field, respectively. From the comparison result in Table 3, we can find that the communication cost of Authentication and Storage Checking is $3l_{\mathbb{G}} + 1l_{\mathbb{Z}}$ and $(4\rho + z + 2)l_{\mathbb{Z}}$, respectively. That is to say, as the number of data blocks increases, the communication cost of Authentication remains unchanged but the communication cost of Storage Checking will be higher.

Table 3. Communication cost of authentication and storage checking.

Phase	**Computational Cost**
Authentication	$3l_{\mathbb{G}} + 1l_{\mathbb{Z}}$
Storage Checking	$(4\rho + z + 2)l_{\mathbb{Z}}$

* $l_{\mathbb{G}}, l_{\mathbb{Z}}$: Time required to perform the corresponding operations.
* ρ: The number of the challenged data blocks.
* z: The total number of data blocks.

The proposed scheme is simulated on the PBC [26] and GMP [27] constructed experimental platform. The experimental platform is configured on a computer with 8GB RAM and Intel 2.60 GHZ CPU. In the simulation, the basis of total number of data blocks and the challenged data blocks is 100 and 50, respectively. From Figure 2, we can see that the computational time at the side of client, Authority and Server increases linearly with the number of data blocks. Moreover, the computational time at Authority side is the least and the computational time at Client side is the most under the condition of the same number of data blocks. Figure 3 is the simulation of computational time at the different entity side. From Figure 3, we can find that the computational time of Storage Checking phase growth linearly with the number of data blocks. The computational time of Authentication is a constant under the different number of data blocks. Hence, it can be determined that the simulation result is consistent with the result of the above theoretical analysis.

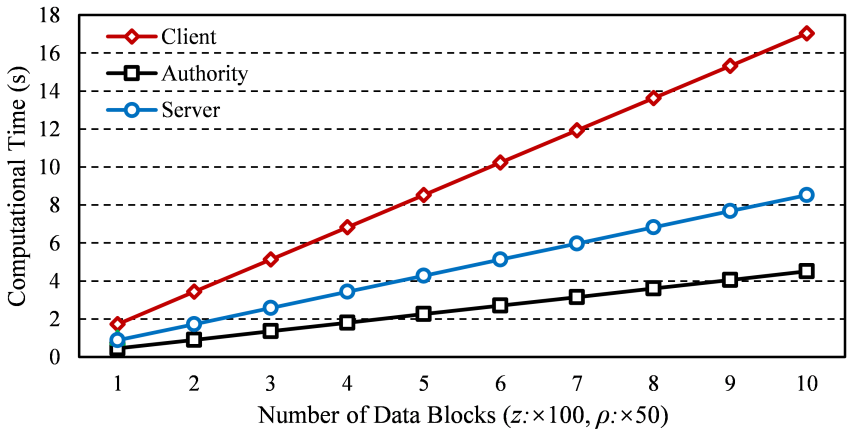

Figure 2. The simulation of computational time at the different entity side.

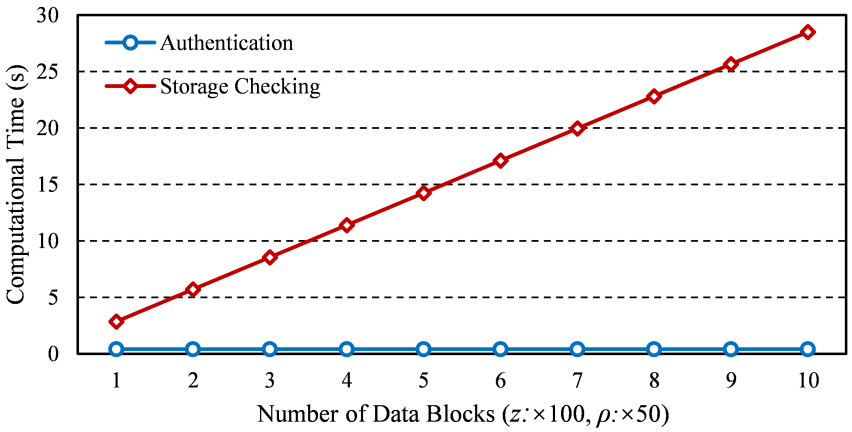

Figure 3. The simulation of computational time of authentication and storage checking.

6. Conclusions

To enhance the security of server access and data storage, in this paper, we propose a secure mutual authentication scheme for cloud computing to implement the authentication between clients and cloud servers. The technique of chameleon hash signature is used to design the authentication mechanism, which makes the designed authentication have a variety of security performance such as tamper attack and replay attack resistance. Moreover, the proposed scheme supports data storage checking for clients. To reduce the local computational overhead, the storage checking tasks are delegated to the fully-trusted entity. Security analysis shows that the proposed mechanisms of authentication and storage checking can be proved to be correct. Performance analysis presents that the proposed scheme has a low computational overhead. Hence, the proposed scheme can be well used in practical cloud-assisted IoT environment for data storage.

Acknowledgments

This work is supported by the National Science Foundation of China under Grant No. 62102169, No. 62072249, the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant 21KJB520033, the Key Research and Development Program (Social Development) of Lianyungang under Grant SF2102, the Research Start-up Fund of JOU under Grant KQ20039.

Conflict of interest

The authors declare no conflict of interest.

References

1. S. Lins, P. Grochol, S. Schneider, A. Sunyaev, Dynamic certification of cloud services: Trust, but verify!, *IEEE Secur. Priv.*, **14** (2016), 66–71. https://doi.org/10.1109/MSP.2016.26
2. T. Menouer, N. Sukhija, P. Darmon, Towards a parallel constraint solver for cloud computing environments, in *Int. Conference Big Data Comput. Serv. Appl.*, (2019), 195–198. https://doi.org/10.1109/BigDataService.2019.00033
3. L. Zhang, Z. Zou, W. Wang, Z. Jin, Y. Su, H. Chen, Resource allocation and trust computing for blockchain-enabled edge computing system, *Comput. Secur.*, **105** (2021), 102249. https://doi.org/10.1016/j.cose.2021.102249
4. M. Hossain, R. Khan, S. A. Noor, R. Hasan, Jugo: A generic architecture for composite cloud as a service, in *2016 IEEE Int. Conf. Cloud Comput.*, (2016), 806–809. https://doi.org/10.1109/CLOUD 2016.0112
5. D. Liu, J. Shen, A. Wang, C. Wang, Secure real-time image protection scheme with near-duplicate detection in cloud computing, *J. Real-Time Image Process.*, **17** (2020), 175–184. https://doi.org/10.1007/s11554-019-00887-6
6. Y. Ren, L. Yan, Y. Cheng, W. Jin, Secure data storage based on blockchain and coding in edge computing, *Math. Biosci. Eng.*, **16** (2019), 1874–1892. https://doi.org/10.3934/mbe.2019091
7. D. Liu, Y. Zhang, D. Jia, Q. Zhang, X. Zhao, H. Rong, Toward secure distributed data storage with error locating in blockchain enabled edge computing, *Comput. Stand. Interfaces*, **79** (2022), 103560. https://doi.org/10.1016/j.csi.2021.103560
8. L. Zhang, Z. Zhang, W. Wang, Z. Jin, Y. Su, H. Chen, Research on a covert communication model realized by using smart contracts in blockchain environment, *IEEE Syst. J.*, (2021). https://doi.org/10.1109/JSYST.2021.3057333
9. W. Wang, H. Huang, L. Zhang, C. Su, Secure and efficient mutual authentication protocol for smart grid under blockchain, *Peer Peer Netw. Appl.*, **14** (2020), 2681–2693. https://doi.org/10.1007/s12083-020-01020-2
10. D. Liu, Y. Zhang, W. Wang, K. Dev, S. A. Khowaja, Flexible data integrity checking with original data recovery in IoT-enabled maritime transportation systems, in *IEEE Trans. Intell. Transp. Syst.*, (2021). https://doi.org/10.1109/TITS.2021.3125070

11. Y. Li, L. Du, G. Zhao, J. Guo, A lightweight identity-based authentication protocol, in *IEEE Int. Conf. Signal Process.*, (2013), 1–4. https://doi.org/10.1109/ICSPCC.2013.6664134

12. Y. Tian, G. Chen, J. Li, A new ultralightweight rfid authentication protocol with permutation, *IEEE Commun. Lett.*, **16** (2012), 702–705. https://doi.org/10.1109/LCOMM.2012.031212.120237

13. X. Li, Y. Han, J. Gao, J. Niu, Secure hierarchical authentication protocol in VANET, *IET Inf. Secur.*, **14** (2020), 99–110. https://doi.org/10.1049/iet-ifs.2019.0249

14. S. P. Shieh, W. Yang, H. Sun, An authentication protocol without trusted third party, *IEEE Commun. Lett.*, **1** (1997), 87–89.

15. G. Ateniese, M. Steiner, G. Tsudik, New multiparty authentication services and key agreement protocols, *IEEE J. Sel. Areas Commun.*, **18** (2000), 628–639. https://doi.org/10.1109/49.839937

16. H. Li, Y. Dai, T. Ling, H. Yang, Identity-based authentication for cloud computing, in *IEEE Int. Conf. Cloud Comput.*, (2009), 157–166. https://doi.org/10.1007/978-3-642-10665-1_14

17. A. J. Choudhury, P. Kumar, M. Sain, H. Lim, H. J. Lee, A strong user authentication framework for cloud computing, *Proc. IEEE Asia-Pacific Serv. Comput. Conf.*, (2011), 110–115. https://doi.org/10.1109/APSCC.2011.14

18. H. Liu, H. Ning, Q. Xiong, L. Yang, Shared authority based privacy-preserving authentication protocol in cloud computing, *IEEE Trans. Parallel Distrib. Syst.*, **26** (2014), 241–251. https://doi.org/10.1109/TPDS.2014.2308218

19. S. C. Patel, R. S. Singh, S. Jaiswal, Secure and privacy enhanced authentication framework for cloud computing, *Int. Conf. Electron. Commun. Syst.*, (2015), 1631–1634. https://doi.org/10.1109/ECS.2015.7124863

20. D. Liu, J. Shen, A. Wang, C. Wang, Lightweight and practical node clustering authentication protocol for hierarchical wireless sensor networks, *Int. J. Sensor Networks*, **27** (2018), 95–102. https://doi.org/10.1504/IJSNET.2018.092638

21. S. Ruj, M. Stojmenovic, A. Nayak, Decentralized Access Control with Anonymous Authentication of Data Stored in Clouds, *IEEE Trans. Parallel Distrib. Syst.*, **25** (2014), 384–394. https://doi.org/10.1109/TPDS.2013.38

22. S. Chandrasekhar, M. Singhal, Efficient and scalable query authentication for cloud-based storage systems with multiple data sources, *IEEE Trans. Serv. Comput.*, **10** (2017), 520–533. https://doi.org/10.1109/TSC.2015.2500568

23. J. L. Tsai, N. W. Lo, A privacy-aware authentication scheme for distributed mobile cloud computing services, *IEEE Syst. J.*, **9** (2017), 805–815. https://doi.org/10.1109/JSYST.2014.2322973

24. X. Liu, Y. Xia, S. Jiang, F. Xia, Y. Wang, Hierarchical attribute-based access control with authentication for outsourced data in cloud computing, *J. Phys. Condens. Matter*, (2013), 477–484. https://doi.org/10.1109/TrustCom.2013.60

25. R. Amin, N. Kumar, G. P. Biswas, R. Iqbal, V. Chang, A light weight authentication protocol for iot-enabled devices in distributed cloud computing environment, *Future Gener. Comput. Syst.*, **78** (2016), 1005–1019. https://doi.org/10.1016/j.future.2016.12.028

26. D. Boneh, Identity-based encryption from the weil pairing, In *Annual international cryptology conference*, Springer, Berlin, Heidelberg, 2001, 213–229. https://doi.org/10.1007/3-540-44647-8_13

27. H. Krawczyk, T. Rabin, Chameleon signatures, in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA*, (2000), 143–154.

28. T. Khan, K. Singh, M. H. Hasan, K. Ahmad, G. T. Reddy, S. Mohan, et al., ETERS: A comprehensive energy aware trust-based efficient routing scheme for adversarial WSNs, *Future Gener. Comput. Syst.*, **125** (2021), 921–943. https://doi.org/10.1016/j.future.2021.06.049

29. W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, C. Su, Blockchain-based reliable and efficient certificateless signature for IIoT devices, *IEEE Trans. Industr. Inform.*, (2021), 1–9. https://doi.org/10.1109/TII.2021.3084753

30. S. Guo, D. Zeng, Y. Xiang, Chameleon hashing for secure and privacy-preserving vehicular communications, *IEEE Trans. Parallel Distrib. Syst.*, **25** (2014), 2794–2803. https://doi.org/10.1109/TPDS.2013.277

31. D. Liu, J. Shen, P. Vijayakumar, A. Wang, T. Zhou, Efficient data integrity auditing with corrupted data recovery for edge computing in enterprise multimedia security, *Multimed. Tools. Appl.*, **79** (2020), 10851–10870. https://doi.org/10.1007/s11042-019-08558-1

32. J. Zhang, X. Zhao, Efficient chameleon hashing-based privacy-preserving auditing in cloud storage, *Cluster Comput.*, **19** (2016), 47–56. https://doi.org/10.1007/s10586-015-0514-0

AIMS Press