SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2017, Vol. 60 122104:1–122104:13
doi: 10.1007/s11432-016-0038-6

Provably secure cloud storage for mobile networks with less computation and smaller overhead

Rui ZHANG[1], Hui MA[1,2*], Yao LU[3] & Yang LI[4]

[1]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093*, China;*
[2]*University of Chinese Academy of Sciences, Beijing* 100049*, China;*
[3]*Department of Complexity Science and Engineering, University of Tokyo, Kashwa City, Japan;*
[4]*Oracle, Beijing* 100193*, China*

Received August 18, 2016; accepted September 18, 2016; published online April 27, 2017

Abstract Secure cloud storage (SCS) guarantees the data outsourced to the cloud to remain intact as it was before being outsourced. Previous schemes to ensure cloud storage reliability are either computationally heavy or admitting long overheads, thus are not suitable for mobile networks with strict computation/bandwidth restrictions. In this paper, we build an efficient SCS system for mobile networks based on homomorphic MAC and propose domain extension to enhance the security level and flexibility of the system. In addition, we give a formal security model which is compatible to previous ones and analyze our system in that model. We also give implementations on mobile devices to verify the effectiveness of our system.

Keywords secure cloud storage, mobile networks, homomorphic MAC, domain extension, applied cryptography

Citation Zhang R, Ma H, Lu Y, et al. Provably secure cloud storage for mobile networks with less computation and smaller overhead. Sci China Inf Sci, 2017, 60(12): 122104, doi: 10.1007/s11432-016-0038-6

1 Introduction

Secure cloud storage. Mobile networks, such as 3G/4G, vehicular ad hoc network (VANET) and Internet of Things (IoT), generate tremendous amount of data, which are transferred through networks and eventually stored in cloud servers. After outsourcing the data to cloud servers, owner of the data, may want to retrieve the data when necessary. However, availability of these data is unsure, as the actual data is under the control of the cloud storage providers, and the users do not have a copy of the data themselves. As a matter of fact, even the best-known storage-service, Amazon's S3, has ever experienced large-scale downtime[1)]. Even worse, to save investments on storage equipments, a cloud storage provider may also be financially motivated to "lose" the user data. Therefore, designing a secure cloud storage (SCS) system with a reliable audit mechanism for mobile networks becomes very important, and has attracted much attention of the researchers.

* Corresponding author (email: mahui@iie.ac.cn)
1) Amazon, http://status.aws.amazon.com/s3us-20080720.html and http://status.aws.amazon.com/s3-20080720.html.

Related work. Deswarte, Quisquater and Saïdane [1], Filho and Barreto [2], Naor and Rothblum [3], and Schwarz and Miller [4] were among the first to consider the problem of remotely checking data integrity. Later research focused designing schemes with formal security models, e.g., authenticators [3], PDP [5–7] and proof of retrievability (PoR) [8–10]. Subsequently, refs. [11–13] focused on the improvement of communication efficiency, refs. [7, 14–16] proposed dynamic PDP schemes, refs. [17–19] supported efficient PoR with dynamic update, and ref. [20] proposed a symmetric-key based PoR scheme with public verifiability. Up to now, all the known PDP schemes still could not conquer two efficiency obstacles, i.e., overhead of data transmission during an audit, and computation cost during an audit. Let us briefly review them below.

First, the large data transmission overhead usually comes from the authentication header, e.g., message authentication codes (MACs) [3, 8], signatures with public verifiability [9, 21–24]. On the other hand, the online computation cost during an audit usually comes from commutations from pubic key cryptography [9, 21, 22, 24]: a typical digital signature, say, one RSA modular exponentiation computation is at least a few dozen to a few hundreds of times slower than a computation of hash functions.

Consider the following example in a cloud logging system for a VANET where a message itself may be very short, say, 16-bit indicator messages. Suppose the authentication headers are implemented by a standard HMAC-SHA1 construction and take the number of messages in a road journey for an audit as $t = 10^4$. The communication from the root node alone to the base station is then $(160 + 16) \cdot t = 1.76$ $\times 10^6$ bits. In other words, the auxiliary data consumes more computation resources and the bandwidth. We remark that it is still much work to do towards an efficient SCS system that is suitable for mobile networks.

Intuition for our work. Our solution is based on a new tool called homomorphic MAC [25, 26] with many customized optimizations in the system design. As a result, we are able to have an SCS system with good performances. A homomorphic MAC is a special type of MAC, s.t., given a set of messages and the corresponding MACs, there is a "combine" algorithm that outputs a linear combination of the messages and the corresponding MACs. The security requirement for a homomorphic MAC is that no probabilistic polynomial time (PPT) adversary can output linear combination of the new data that was not authenticated by the user. Homomorphic MAC was first introduced for preventing pollution attacks in network coding [25].

In our SCS system, we use homomorphic MACs to "compress" the headers for different data blocks into one. It is not hard to see that this construction also provides extractability of data, since in the execution of the audit protocol a (homomorphic) MAC cannot be verified without the actual message. Moreover, we remove the coefficient matrix $\boldsymbol{E}$ from the data block and use id to identify the file. Since a homomorphic MAC only uses symmetric key cryptography, it is much more efficient. In this way, we are able to achieve less computation with smaller overhead for data transmission and storage.

Our contributions. Firstly, we propose an efficient SCS scheme and give the security proof in the standard model. Secondly, we propose domain extension technique to enhance the security level and flexibility of our scheme. The above two schemes both achieve less computation with smaller overhead for data transmission and storage. Consider the previous example of the cloud logging system for a VANET, if we take homomorphic MAC with output of 160 bits, the total data to transmit becomes $16 \cdot t + 160 \approx 1.6 \times 10^4$ bits. Our scheme can reduce the storage to 1% and the bandwidth overhead to 0.01%! Finally, we give the first efficient SCS implementation on the mobile device, e.g., for a file of 24.9 MB on a mobile device, the time for outsourcing is only 1.8 s, the time for a verification is only 0.2 s. We also compare it with some well-known work [5, 9, 10, 24]. After careful analysis, we conclude that our SCS system is efficient and practical.

2 Preliminary

In this section, we review some useful notations and definitions.

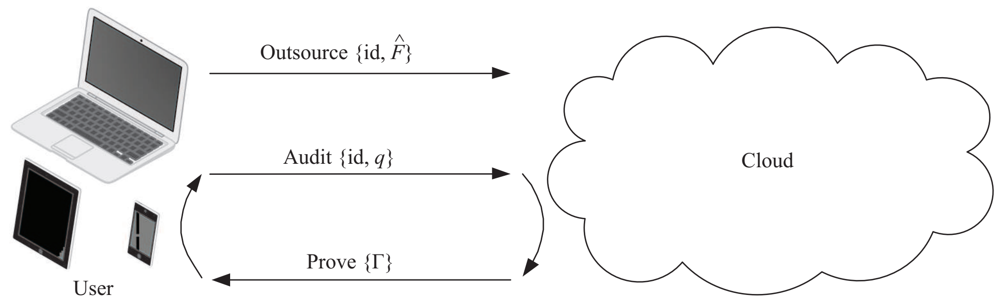

Figure 1 The system model.

Notations. If x is a string, let $|x|$ denote its length, while if S is a set then $|S|$ denotes its size. If S is a set then $s \leftarrow S$ denotes the operation of picking an element s of S uniformly at random. We write $z \leftarrow \mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $(x, y, \ldots)$ and an output z. If $k \in \mathbb{N}$, a function $f(k)$ is negligible if $\exists\ k_0 \in \mathbb{N}$, $\forall\ k > k_0$, $f(k) < 1/k^c$, where $c > 0$ is a constant. We say an algorithm is probabilistic polynomial time (PPT) if it uses randomness (i.e, flips coins) and its running time is bounded by some polynomial in the input size (or often a polynomial in a security parameter).

Definition 1 (Pseudorandom generator (PRG))**.** A (t, ϵ)-PRG is a function $G : \{0,1\}^n \to \{0,1\}^m (m \gg n)$, s.t., G is computable by a deterministic algorithm within polynomially bounded time. Furthermore, $\forall$ t-time algorithms $\mathcal{A}$,

$$\Pr[\mathcal{A}(G(S)) = 1 | S \leftarrow \{0,1\}^n] - \Pr[\mathcal{A}(R) = 1 | R \leftarrow \{0,1\}^m] \leqslant \epsilon. \tag{1}$$

We say that a PRG is (t, ϵ)-secure, if the uniform distribution on $\{0,1\}^m$ is (t, ϵ)-indistinguishable from distributions $\{G(S) | S \leftarrow \{0,1\}^n\}$. In particular, we say a PRG is secure, if $\forall$ polynomially bounded t, ϵ is negligible.

Definition 2 (Pseudorandom Function (PRF))**.** A (t, ϵ, q)-PRF is a function $f_K : X \times K \to M$ (where $X \in \{0,1\}^n, K \in \{0,1\}^s, M \in \{0,1\}^m$), if the following conditions are satisfied: (1) Given any $X \in \{0,1\}^n$ and $K \in \{0,1\}^s$, there always exists a PPT algorithm to compute f_K. (2) For any t-time algorithm $\mathcal{A}$, there is

$$|\Pr[\mathcal{A}^{f_K} = 1 | K \leftarrow \{0,1\}^s] - \Pr[\mathcal{A}^f = 1 | f \in \mathcal{F}]| \leqslant \epsilon, \tag{2}$$

where $\mathcal{F} = \{f : \{0,1\}^n \to \{0,1\}^m\}$ and $\mathcal{A}$ makes at most q queries to the oracle. In particular, we say a PRF is secure, if $\forall$ PPT adversary $\mathcal{A}$, and $\forall$ polynomially bounded q, ϵ is negligible.

3 Secure cloud storage system

Security considerations towards SCS security can usually be classified into two categories: the system aspect and the cryptography aspect. In the system aspect, (1) An SCS system is required to be efficient and the storage overhead on the server should be small. (2) The number of supported audit process should not be limited to a fixed bound. (3) The audit should be stateless, i.e., the verifier should not need to maintain some state information to complete the audit. In the cryptography aspect, one wants to make sure that if a storage server passes the verification for a file, no matter that it could be a malicious server with Byzantine behavior, it should actually store the file at the moment of the audit. However, if one wants to interpret this intuition into a formal security model and an implementation with satisfactory performances and rigorous security analysis, the problem becomes extremely difficult.

3.1 The system model

The model of an SCS system is depicted in Figure 1. First, a user outsources its data to a cloud, then the user confirms the data integrity by interacting with the cloud using an SCS protocol. The motivation of such integrity check is that the user's data could be modified without the acknowledgement of the user. The user periodically performs an audit on the data integrity, and the cloud returns a proof for the corresponding audit. The user can then check whether the proof is valid or not. If it is valid, it means that the data remains intact, otherwise, means that the data has been tampered. The precise description of an SCS system is as follows.

Definition 3 (Secure cloud storage (SCS))**.** An SCS scheme consists of five polynomial algorithms (KeyGen, Outsource, Audit, Prove, Verify).

• $K \leftarrow \mathrm{KeyGen}(\lambda)$. On input a security parameter λ, the user runs this algorithm to generate a secret key K to enable verification.

• $(\mathrm{id}, \hat{F}) \leftarrow \mathrm{Outsource}(F, K)$. On input the data file F to be outsourced, the user runs this algorithm to produce an unique identifier id and the processed data file $\hat{F}$ using the secret key K. The user sends the file $\hat{F}$ together with its identifier to the cloud.

• $(q, \mathrm{id}) \leftarrow \mathrm{Audit}(K)$. The user runs this algorithm to generate an audit query q and an identifier id which will be sent to the cloud.

• $\Gamma \leftarrow \mathrm{Prove}(q, \mathrm{id}, \hat{F})$. On input an audit query q and an identifier id, the cloud computes a proof Γ using the stored data $\hat{F}$.

• $\delta \leftarrow \mathrm{Verify}(q, \Gamma, \mathrm{id}, K)$. On input an audit query q, an identifier id and the cloud's proof Γ, the user checks whether the cloud's proof is correct using the secret key K. The user outputs $\delta = 1$ if the proof is correct, else outputs $\delta = 0$.

3.2 The security model

We present security definitions regarding an SCS system. Suppose that in an SCS system, only the data owner is trusted and the cloud server is untrusted as in [5], namely, it must answer challenges from the client, even though the file is totally or partially missing, the server may try to convince the client that it possessed the file. We insist that an SCS scheme should simultaneously satisfy correctness and soundness conditions.

Correctness captures the intuition that "correct data" will always pass the verification process. It requires that, $\forall$ key K outputted by KeyGen, $\forall\ F \in \{0,1\}^*$, and $\forall\ (\hat{F}, \Gamma)$ outputted by $\mathrm{Prove}(q, \mathrm{id}, \hat{F}) \rightarrow \Gamma$, the verification algorithm accepts when interacting with the valid prover:

$$(\mathrm{Verify}(q, \Gamma, \mathrm{id}, K) \Leftrightarrow \mathrm{Prove}(q, \mathrm{id}, \hat{F})) = 1. \tag{3}$$

Soundness means that the corrupted data should be detected by the verification algorithm with overwhelming probability. Formally, the security game between a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a PPT challenger $\mathcal{C}$, an SCS scheme $\Phi = (\mathrm{KeyGen}, \mathrm{Outsource}, \mathrm{Audit}, \mathrm{Prove}, \mathrm{Verify})$ is as follows.

• **Setup.** The challenger $\mathcal{C}$ runs KeyGen to obtain the secret key K.

• **Learning.** The adversary $\mathcal{A}$ adaptively makes poly(λ) number of queries where a query is one of the following:

- Outsource query. Given a data file F chosen by $\mathcal{A}$, the challenger $\mathcal{C}$ responses by running Outsource algorithm and sending the $\hat{F}$ and id to $\mathcal{A}$.

- Verification query. Given a file identifier id chosen by $\mathcal{A}$, if id is the output of some previous outsource query that $\mathcal{A}$ has made, then the challenger $\mathcal{C}$ initiates a verification with $\mathcal{A}$, the data file F associated to the identifier id in this way: $\mathcal{C}$ chooses a random challenge q by running Audit algorithm; $\mathcal{A}$ produces a proof Γ, the challenge q; $\mathcal{C}$ verifies the proof Γ by running Verify algorithm.

In the end $\mathcal{C}$ sends the decision bit δ to $\mathcal{A}$ as feedback. Otherwise, if id is not the output of any previous outsource query that $\mathcal{A}$ has made, $\mathcal{C}$ does nothing.

• **Cheat.** The adversary $\mathcal{A}$ outputs a Γ^* for a query q^* (chosen by $\mathcal{C}$) and an identifier id* chosen from all file identifiers it obtains from $\mathcal{C}$ by asking outsource queries during **Learning** phase, where Γ^* is not computed using the processed data file F^* which accessed by asking outsource queries, and the query q^* is not asked before. The adversary $\mathcal{A}$ wins the game if and only if the challenge $\mathcal{C}$ accepts this proof.

We denote the above game as $\mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)$. Soundness requires that

$$\Pr[\text{Soundness}] = 1 - \Pr[\mathcal{A} \text{ wins } \mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)] \geqslant 1 - \epsilon(\lambda), \tag{4}$$

where

$$\Pr[\mathcal{A} \text{ wins } \mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)] \leqslant \epsilon(\lambda) \tag{5}$$

is a negligible function of a security parameter λ.

Besides, the user wants to extract its data from the audit queries and the cloud's proof if the verification algorithm accepts the proof of the prover. It is convenient for the user to verify and download the data at the same time. We define the security games in a similar way as in [24]. Formally, the security game between a probabilistic polynomial time (PPT) adversary $\mathcal{B}$ and a PPT challenger $\mathcal{C}$ is as follows.

- **Setup.** The challenger $\mathcal{C}$ runs the key generation algorithm KeyGen to obtain the secret key K.
- **Learning.** This phase is the same as the **Learning** phase of the game $\mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)$.
- **Commit.** The adversary $\mathcal{B}$ chooses a file identifier id^* among all file identifiers she obtains from $\mathcal{C}$ by asking outsource queries during **Learning** phase, and commits id^* to $\mathcal{C}$. Let F^* denote the data file associated to identifier id^*.
- **Extract.** The challenger $\mathcal{C}$ initiates η verifications with $\mathcal{B}$, the data file F^*, where $\mathcal{C}$ plays the role of verifier and $\mathcal{B}$ plays the role of prover, as during the **Learning** phase. From messages collected in these η interactions with $\mathcal{B}$, $\mathcal{C}$ extracts a data file F' using a PPT extractor algorithm. The adversary B wins this game, if and only if $F' \neq F^*$.

The adversary $\mathcal{B}$ is ϵ-admissible, if the probability that $\mathcal{B}$ convinces $\mathcal{C}$ to accept F' in a verification during the **Extract** phase, is at least ϵ. We denote the above game as $\mathrm{Game}_{\mathcal{B}}^{\Phi}(\eta)$. It requires

$$\Pr[\mathcal{B} \text{ wins } \mathrm{Game}_{\mathcal{B}}^{\Phi}(\eta)] \leqslant \epsilon(\lambda) \tag{6}$$

is a negligible function of a security parameter λ.

Definition 4 (Secure SCS)**.** An SCS scheme is secure, if (1) it is correct; (2) $\Pr[\mathcal{A} \text{ wins } \mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)]$ is negligible, and (3) for any PPT ϵ-admissible adversary $\mathcal{B}$ with non-negligible ϵ, there exists a polynomial η, such that the advantage $\mathrm{Adv}_{\mathcal{B}}^{\Phi}(\eta)$ defined as follows is negligible:

$$\mathrm{Adv}_{\mathcal{B}}^{\Phi}(\eta) = \Pr[\mathcal{B} \text{ wins } \mathrm{Game}_{\mathcal{B}}^{\Phi}(\eta)]. \tag{7}$$

The intuition of the above definition is that no PPT adversary $\mathcal{B}$ can interact with a user twice with identifier id^*, and a user will be able to extract two distinct files (F^*, F') after two successful verifications. In other words, it is hard to create one authentication header that works for two files.

It is easy to see that the above security model is "compatible" with major previous work [5,9], namely, they provide similar security guarantees.

4 Our scheme and security analysis

In this section, we construct an efficient SCS scheme with private verifiability and analyze its security.

4.1 The proposed scheme

According to Definition 3, the description of our scheme $\Phi = (\mathrm{KeyGen}, \mathrm{Outsource}, \mathrm{Audit}, \mathrm{Prove}, \mathrm{Verify})$ is as follows.

- $K \leftarrow \mathrm{KeyGen}(\lambda)$. The user picks a pseudorandom generator $G : \mathcal{K}_G \to \mathbb{F}_q^n$ and a pseudorandom function $H : \mathcal{K}_F \times (\mathcal{I} \times [m]) \to \mathbb{F}_q$. The secret key is $K = (k_1, k_2)$ where $k_1 \in \mathcal{K}_G$ and $k_2 \in \mathcal{K}_F$.
- $(\mathrm{id}, \hat{F}) \leftarrow \mathrm{Outsource}(F, K)$. Given the file F, the user assigns it an unique identifier id (including the length information), and splits F into a collection of vectors $\{\boldsymbol{v}_i = [v_{i1}, \ldots, v_{in}]\}_{i=1,\ldots,m}$ in $\mathbb{F}_q^n$ (for some n). For each $\boldsymbol{v}_i$, compute its tag using key $K = (k_1, k_2)$ as follows.

  - $\boldsymbol{u} \leftarrow G(k_1) \in \mathbb{F}_q^n$, is a vector consisting of n elements.

- $b_i \leftarrow H(k_2, (\mathrm{id}, i)) \in \mathbb{F}_q$.
- $t_i \leftarrow \langle \boldsymbol{u} \cdot \boldsymbol{v}_i \rangle + b_i$.

The user outsources the processed data $\hat{F} = \{\boldsymbol{v}_i, t_i\}$, where $1 \leqslant i \leqslant m$, together with its identifier id to the cloud.

• $(q, \mathrm{id}) \leftarrow \mathrm{Audit}(K)$. The user runs this algorithm to generate a collection of numbers $\{i_j, c_j\}_{j=1,\ldots,l}$ where $1 \leqslant i_j \leqslant m$ and $c_j \in \mathbb{F}_q$. The user sends the query $q = \{i_j, c_j\}_{j=1,\ldots,l}$ and an identifier id to the cloud.

• $\Gamma \leftarrow \mathrm{Prove}(q, \mathrm{id}, \hat{F})$. On receiving an audit query $(q = \{i_j, c_j\}_{j=1,\ldots,l}, \mathrm{id})$ where l is the length of the query, the cloud first finds $(\boldsymbol{v}_{i_j}, t_{i_j})$ for each queried data blocks. It then computes $\boldsymbol{w} = \sum_{j=1}^{l} c_j \boldsymbol{v}_{i_j}$ and $t = \sum_{j=1}^{l} c_j t_{i_j}$. At last the cloud sends back $\Gamma = (\boldsymbol{w}, t)$ as a proof of the corresponding query.

• $\delta \leftarrow \mathrm{Verify}(q, \Gamma, \mathrm{id}, K)$. On input an audit query $(q = \{i_j, c_j\}_{j=1,\ldots,l}, \mathrm{id})$, the cloud's proof $\Gamma = (\boldsymbol{w}, t)$, let $K = (k_1, k_2)$ be a secret key, the user does the following steps:

- $\boldsymbol{u} \leftarrow G(k_1) \in \mathbb{F}_q^n$.
- $a \leftarrow \langle \boldsymbol{u} \cdot \boldsymbol{w} \rangle \in \mathbb{F}_q$.
- $b \leftarrow \sum_{j=1}^{l} [c_{i_j} \cdot H(k_2, (\mathrm{id}, i_j))] \in \mathbb{F}_q$.

The user checks whether $a + b = t$. If they are equal, the user outputs 1, otherwise outputs 0.

Then we prove the security of our scheme as was given in Definition 4. We omit the proof for the correctness of our scheme here since it is obvious from the descriptions above.

We show a malicious cloud cannot cheat with overwhelming probability, i.e., $\Pr[\mathcal{A} \text{ wins } \mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)]$ is negligible. In particular, we have the following theorem.

Theorem 1. Assume that G is a secure PRG and H is a secure PRF. For a PRF adversary $\mathcal{B}_1$ we let $\mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{PRF}}$ denote $\mathcal{B}_1$'s advantage in winning the PRF security game with respect to F. Similarly, for a PRG adversary $\mathcal{B}_2$ we let $\mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{PRG}}$ be $\mathcal{B}_2$'s advantage in winning the PRG security game with respect to G. Then

$$\Pr[\mathcal{A} \text{ wins } \mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)] \leqslant \mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{PRF}} + \mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{PRG}} + \frac{1}{q}. \tag{8}$$

Proof. We prove the theorem using a sequence of games [27] which we denote as game 0, 1, 2. For $i = 0, 1, 2$, let W_i be the event that $\mathcal{A}$ wins the game in game i.

Game 0. Game 0 is identical to the actual attack. Therefore,

$$\Pr[W_0] = \Pr[\mathcal{A} \text{ wins } \mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)]. \tag{9}$$

Game 1. In game 1, we replace the output of the PRG used in SCS with a truly random string. Then there is a PRG adversary $\mathcal{B}_2$ such that

$$|\Pr[W_0] - \Pr[W_1]| \leqslant \mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{PRG}}. \tag{10}$$

Game 2. In game 2, we replace the PRF by a truly random function. Then there is a PRF adversary $\mathcal{B}_1$ such that

$$|\Pr[W_1] - \Pr[W_2]| \leqslant \mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{PRG}}. \tag{11}$$

Next we show that $\Pr[W_2] = 1/q$ in game 2. Since the adversary outputs $(q^*, \Gamma^* = (\boldsymbol{v}^*, t^*))$. The adversary wins the game if $\mathrm{Verify}(q^*, \Gamma^*, K) = 1$, which means that

$$t^* = \langle \boldsymbol{u} \cdot \boldsymbol{v}^* \rangle + \sum_{j=1}^{l} [c_{i_j}^* \cdot H(k_2, (\mathrm{id}, i_j))]. \tag{12}$$

Note that the value of $\langle \boldsymbol{u} \cdot \boldsymbol{v}^* \rangle + \sum_{j=1}^{l} [c_{i_j}^* \cdot H(k_2, (\mathrm{id}, i_j))]$ is a random value independent of the adversary's view. Therefore, the probability that adversary wins is exactly $\frac{1}{q}$.

Putting all these together we obtain

$$\Pr[\mathcal{A} \text{ wins } \mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)] \leqslant \mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{PRF}} + \mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{PRG}} + \frac{1}{q}. \tag{13}$$

Since our scheme utilizes the homomorphic MACs as the authenticators, then we can extract the data from the collected results of the interactions with the cloud, if the user accepts the proof of the cloud.

4.2 Domain extension for authentication tags

In practice, the predetermined parameter q is typically set as $2^8 = 256$. Hence, this security level for our scheme may not always be sufficient. Cheng and Jiang [26] proposed an algorithm that employed the trace function over finite fields to construct the tags of the files, which can achieve a reliable security $\frac{1}{q^l}$ $(l \geqslant 1)$. However, their method increases both the communication overhead and the computation complexity of original scheme.

In this section, we present a simple but efficient method to enhance the security of our scheme. The idea behind is to extend the size of tag by making use of a matrix instead of a vector. Our scheme also includes five algorithms: $\Phi^* = (\mathrm{KeyGen}, \mathrm{Outsource}, \mathrm{Audit}, \mathrm{Prove}, \mathrm{Verify})$.

• $K \leftarrow \mathrm{KeyGen}(\lambda)$. It is the same as that in Subsection 4.1 except that the user picks a pseudorandom generator $G : \mathcal{K}_G \to \mathbb{F}_q^{r\cdot n}$ and a pseudorandom function $H : \mathcal{K}_F \times (\mathcal{I} \times [m]) \to \mathbb{F}_q^r$, where r is the number of the tags for one data block.

• $(\mathrm{id}, \hat{F}) \leftarrow \mathrm{Outsource}(F, K)$. It is the same as that in Subsection 4.1 except that for each $\boldsymbol{v}_i$, the user computes its tag as follows.

- $\boldsymbol{U} \leftarrow G(k_1) \in \mathbb{F}_q^{r\cdot n}$, is a matrix of $r \times n$ dimension.
- $\boldsymbol{b}_i \leftarrow H(k_2, (\mathrm{id}, i)) \in \mathbb{F}_q^r$, is a vector consisting of r elements.
- $\boldsymbol{t}_i \leftarrow \boldsymbol{U} \cdot {\boldsymbol{v}_i}^{\mathrm{T}} + \boldsymbol{b}_i$, is a vector of consisting of r elements.

The user outsources $\hat{F} = \{\boldsymbol{v}_i, \boldsymbol{t}_i\}$ and id to the cloud.

• $(q, \mathrm{id}) \leftarrow \mathrm{Audit}(K)$. It is the same as that in Subsection 4.1.

• $\Gamma \leftarrow \mathrm{Prove}(q, \mathrm{id}, \hat{F})$. It is the same as that in Subsection 4.1 except that the cloud computes $\boldsymbol{t} = \sum_{j=1}^{l} c_j \boldsymbol{t}_{i_j}$ and sends back $\Gamma = (\boldsymbol{w}, \boldsymbol{t})$ as a proof of the corresponding query.

• $\delta \leftarrow \mathrm{Verify}(q, \Gamma, \mathrm{id}, K)$. It is the same as that in Subsection 4.1 except that the user does the following steps:

- $\boldsymbol{U} \leftarrow G(k_1) \in \mathbb{F}_q^{r\cdot n}$, is a matrix of $r \times n$ dimension.
- $\boldsymbol{a} \leftarrow \boldsymbol{U} \cdot \boldsymbol{w}^{\mathrm{T}} \in \mathbb{F}_q^r$, is a vector consisting of r elements.
- $\boldsymbol{b} \leftarrow \sum_{j=1}^{l} [c_{i_j} \cdot H(k_2, (\mathrm{id}, i_j))] \in \mathbb{F}_q^r$, is a vector consisting of r elements.

The user checks whether $\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{t}$. If they are equal, the user outputs 1, otherwise outputs 0.

Theorem 2. Assume that G is a secure PRG and H is a secure PRF. For a PRF adversary $\mathcal{B}_1$ we let $\mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{PRF}}$ denote $\mathcal{B}_1$'s advantage in winning the PRF security game with respect to F. Similarly, for a PRG adversary $\mathcal{B}_2$ we let $\mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{PRG}}$ be $\mathcal{B}_2$'s advantage in winning the PRG security game with respect to G. Then

$$\Pr[\mathcal{A} \text{ wins } \mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)] \leqslant \mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{PRF}} + \mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{PRG}} + \frac{1}{q^r}. \tag{14}$$

Proof. We prove the theorem using a sequence of games [27] which we denote as game 0, 1, 2. For $i = 0, 1, 2$, let W_i be the event that $\mathcal{A}$ wins the game in game i.

Game 0. Game 0 is identical to the actual attack. Therefore,

$$\Pr[W_0] = \Pr[\mathcal{A} \text{ wins } \mathrm{Game}_{\mathcal{A}}^{\Phi}(\eta)]. \tag{15}$$

Game 1. In game 1, we replace the output of the PRG used in SCS with a truly random string. Then there is a PRG adversary $\mathcal{B}_2$ such that

$$|\Pr[W_0] - \Pr[W_1]| \leqslant \mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{PRG}}. \tag{16}$$

Game 2. In game 2, we replace the PRF by a truly random function. Then there is a PRF adversary $\mathcal{B}_1$ such that

$$|\Pr[W_1] - \Pr[W_2]| \leqslant \mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{PRG}}. \tag{17}$$

Next we show that $\Pr[W_2] = 1/q^r$ in game 2. Since the adversary outputs $(q^*, \Gamma^* = (\boldsymbol{v}^*, \boldsymbol{t}^*))$. The adversary wins the game if $\mathrm{Verify}(q^*, \Gamma^*, K) = 1$, which means that

$$\boldsymbol{t}^* = \boldsymbol{U} \cdot (\boldsymbol{v}^*)^{\mathrm{T}} + \sum_{j=1}^{l} [c_{i_j}^* \cdot H(k_2, (\mathrm{id}, i_j))]. \tag{18}$$

Note that the value of $\boldsymbol{U}\cdot(\boldsymbol{v}^*)^{\mathrm{T}}+\sum_{j=1}^{l}[c_{i_j}^*\cdot H(k_2,(\mathrm{id},i_j))]$ is a random value independent of the adversary's view. Therefore, the probability that adversary wins is exactly $\frac{1}{q^r}$.

Putting all these together above we obtain

$$\Pr[\mathcal{A}\text{ wins Game}_{\mathcal{A}}^{\Phi}(\eta)]\leqslant \mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{PRF}}+\mathrm{Adv}_{\mathcal{B}_2}^{\mathrm{PRG}}+\frac{1}{q^r}. \tag{19}$$

Since our scheme utilizes the homomorphic MACs as the authenticators, then we can extract the data from the collected results of the interactions with the cloud, if the user accepts the proof of the cloud.

4.3 Extraction and practicality

Extraction. The user can extract the file vectors during the audit phase. If the user wants to extract u file vectors $\boldsymbol{v}_1,\boldsymbol{v}_2,\ldots,\boldsymbol{v}_u$, where $\{\boldsymbol{v}_i=[v_{i1},\ldots,v_{in}]\}_{i=1,\ldots,u}$, the user first prepares u audit queries $(\{j,c_{kj}\}_{j=1,\ldots,u,k=1,\ldots,u},\mathrm{id})$, where all the c_{kj} are pairwise distinct and the matrix

$$\begin{pmatrix} c_{11} & \cdots & c_{1u} \\ c_{21} & \cdots & c_{2u} \\ & \ddots & \\ c_{u1} & \cdots & c_{uu} \end{pmatrix} \tag{20}$$

is full rank. So we can get u linear equations for $v_{11},v_{21},\ldots,v_{u1}$, $w_{k1}=c_{k1}v_{11}+c_{k2}v_{21}+\cdots+c_{ku}v_{u1}$, where $k=1,\ldots,u$. Since the above matric is full rank, the user can solve these equations (using Gaussian elimination) to obtain $v_{11},v_{21},\ldots,v_{u1}$, and the time complexity of Gaussian elimination is $O(u^3)$. The user can also use the same method to obtain $v_{1a},v_{2a},\ldots,v_{ua}$, where $a=2,\ldots,n$.

Practicality. Note that in most cases, the security parameter $q=256$ is sufficient for many SCS applications, especially for the mobile devices. Let k be the total number of audit queries during an audit. If the cloud does not possess the data, the protocol cannot detect the cheating behavior with probability less than $\frac{1}{q^k}$, which is exponentially small with respect to k.

However, if an 8-bit authentication tag is not sufficient for some applications, we suggest the enhanced scheme with domain extension. The tag size for one data block can be set to any length which satisfies the security need by setting the security parameter r. Along with the tag size increases, the computation cost and communication cost will increase theoretically. The experiment results will be presented in the next section.

5 Implementation and analysis

In this section, we implement our scheme on a mobile phone and a laptop respectively, then compare it with some previous proposals. The source code of our system is available from the authors and the test files we used are public data set2), Gentoo mirrors under the path "gentoo/distfiles/".

5.1 Our implementation

Our implementation is depicted in the Figure 2. The user side is running Android 4.4.4 KTU84P system on a MI 4 mobile phone with a Snapdragon 801 CPU @2.5 GHz and 3 GB RAM. The cloud side is deployed by a hadoop cluster (v2.6.0) which has one master node and three slave nodes, all on a virtual machine VMware (v10.0). For simplicity, the configuration of each node is the same as the user side. Besides the benchmarks on a mobile phone, we also get the benchmarks for the user side on a laptop that is running OS X Yosemite 10.10.1 system on a Macbook Air with an Intel Core i5 CPU @1.3 GHz and 4 GB RAM.

2) Gentoo Mirrors. www.gentoo.org/main/en/mirrors2.xml. 2015.

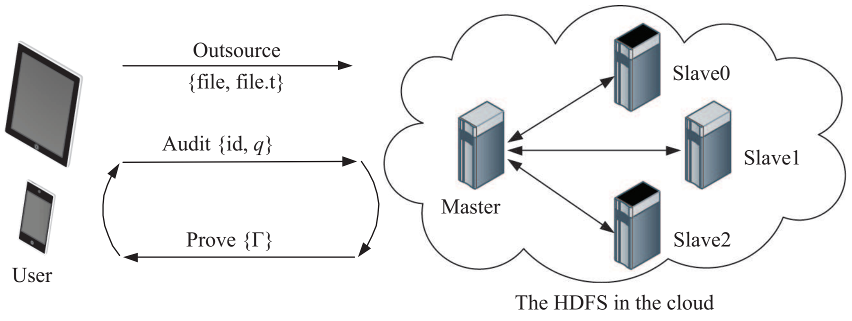

Figure 2 Our implementation.

Table 1 Gentoo data set

Benchmark	File name	File size
#1	0.3.12.tar.gz	23.1 KB
#2	BLT2.4z.tar.gz	2.1 MB
#3	ACE+TAO-5.7.2.tar.bz2	24.9 MB
#4	textures-406files-7-16-04.zip	120 MB
#5	fate-0.8.2.tar.xz	422 MB

We implement the user side in C and Java and implement the cloud in C. In our implementation, we choose $q = 256$, i.e., we work in the finite filed $\mathbb{F}_{2^8}$. For the finite field operations, we choose the tool py_ecc[3] to generate all the results and store them in RAM. According to the size of the file, the user can choose the value of n flexibly, i.e., $n = 1024$ or $n = 1024 \times 1024$. The minimum size of a block that can be used to generate one tag is 8 bits, this provides a fine-grained audit mechanism. We implement the pseudorandom function $\mathcal{H}$ using CBC-AES and the pseudorandom generator $\mathcal{G}$ using KDF [28], which are all from OpenSSL[4]. The cloud utilizes libhdfs[5], a JNI based C API for Hadoop's Distributed File System (HDFS), to upload and download data. As commonly done in practice, we use Socket and SSL to set up a secure channel that transforms information between the user and the cloud.

Besides the original algorithms, we add Server&Client, an algorithm to build the secure channel. Server&Client, KeyGen, Outsource, Audit and Verify algorithms are executed by the user, Server&Client and Prove algorithms are executed by the cloud. The cloud stores the original files and the tag files while the user only stores the secret key.

5.2 Performance evaluation and analysis

In order to demonstrate a complete view on the performance of the system, we present the detailed experimental results. First, we implement the system on a mobile phone and a laptop respectively. We use Gentoo mirrors as the test files that can be found at many websites. The actual data set is reviewed in Table 1. We measure the performance in terms of computation cost, communication cost and storage cost. Next, we compare our system with some existing protocols. We only do the theoretical comparisons due to the absence of their source codes.

Computation cost. The computation cost of our system consists of four proportions, namely, the time for outsourcing, auditing, proving and verifying the data. The results of our implementation on a mobile device are presented in Table 2. Outsourcing the file occupies more time than other functions. The time for outsourcing is influenced by the actual file size. The data set #5 of 422 MB takes the maximal 26.6 s among the test files. For the data set #3 of 24.9 MB, when N is set to 1 K, it takes about 2.0 s, which is close to 1.7 s when N is set to 1 M, because when the file size is fixed, the total computation cost maintains steady no mater what N is. We suggest that when large files arrive, N is set to 1 M rather than 1 K because this can decrease the length of the proof and the storage cost. The

3) Martinian E. Erasure correcting codes in python. web.mit.edu/emin/www.old/index.html. 2014.
4) OpenSSL Cryptography and SSL/TLS Toolkit. http://www.openssl.org/. 2014.
5) C API libhdfs. hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/LibHdfs.html. 2014.

Table 2 Performance of our system on a mobile phone

Benchmark	N	M	Outsource (ms)	Audit (ms)	Prove (ms)	Verify (ms)	Storage	Comm.
#1	1 K	24	2.79	0.01	0.13	1.00	24 B	8 bits
#2	1 K	2150	142.24	0.17	4.95	1.69	2.1 KB	8 bits
#3	1 K	25500	2004.21	0.10	4.02	1.42	24.9 KB	8 bits
#3	1 M	25	1781.66	0.08	91.40	226.88	25 B	8 bits
#4	1 M	121	7881.86	0.04	472.79	227.69	121 B	8 bits
#5	1 M	423	26616.17	0.10	1633.80	210.30	423 B	8 bits

Table 3 Performance of our system on a laptop

Benchmark	N	M	Outsource (ms)	Audit (ms)	Prove (ms)	Verify (ms)	Storage	Comm.
#1	1 K	24	0.63	0.12	0.16	0.12	24 B	8 bits
#2	1 K	2150	26.00	0.19	4.06	0.26	2.1 KB	8 bits
#3	1 K	25500	274.07	0.16	4.64	0.29	24.9 KB	8 bits
#3	1 M	25	269.21	0.10	101.02	54.77	25 B	8 bits
#4	1 M	121	1114.02	0.11	487.62	61.09	121 B	8 bits
#5	1 M	423	3761.53	0.11	1710.99	54.27	423 B	8 bits

time for auditing can be neglected because it is extremely short. In our system, if the file is divided into more than 1000 blocks, the length of the audit query is set to 500, otherwise, it is set to half of the total number of blocks. The time for a proof depends on the block size and the number of audit queries (we ignore the time for downloading a copy of the file from HDFS), the maximal time is 1.6 s for the data set #5, this is fast because Prove is done by the cloud. The time for a verification is influenced by the block size and the number of audit queries, the maximal time is about 2.3 s.

The experiment results on a laptop are shown in Table 3. All results are much faster than the results on a mobile device in Table 2. Specifically, the time for outsourcing is 4.5–7.3 times faster than that on a mobile device, the time for auditing is so short that we can ignore it, the time for proving is close to that on a mobile device because they are done by the same cloud, the time for verifying is 3.7–8.3 times faster than that on a mobile device. Apparently, our implementation is extremely efficient and can satisfy many applications in mobile networks.

Storage cost. A user only keeps the secret key which contains one 256-bit PRF key and one 256-bit PRG key. The cloud needs to store the original file and the authentication tags. The additional storage cost depends on the authentication tags. The experiment results of the storage cost in Tables 2 and 3 are the same. Here, the tag size of one block is 8 bits, and the additional storage cost grows linearly with the total number of blocks. To reduce the storage cost, we can increase the block size adaptively, i.e., the storage cost of the data set #3 decreases from 24.9 KB to 25 B when the block size is increased to 1 MB. The storage cost in our system is always very small. For example, the maximal storage cost is only 24.9 KB of the data set #3. When the block size N is set to 1 K, the additional storage cost is about 0.1% of the original data size. When N is chosen to 1 M, it is about 10^{-6} of the original data size.

Communication cost. The communication cost in our system can be measured by the user side and the cloud side. For the user, the communication cost depends on the length of the audit query, which is flexible to change. For the cloud, the communication cost contains a linear combination of the queried blocks and a linear combination of the queried authentication tags. The former is the same as the block size, and the latter depends on the size of an authentication tag, which decided by the system parameter. Due to the block size is fixed, we only focus on the additional authentication tag. As can be seen from Tables 2 and 3, the additional communication cost is only 8 bits that is the size of one tag.

Private verification vs. public verification. We also compare the performance of our system with the existing efficient public key scheme [24]. We download the source code and reproduce the experiment with our data set, the user side is running Ubuntu 14.04 LTS 64-bit system on a laptop with an Intel Core i5-3320 M CPU @2.6 GHz and 4 GB RAM. In particular, for the data set #4, we set $N = 1$ M, the time for outsourcing is 4.73 h, the time for auditing is 6.4 s, the time for proving is 3.6

Table 4 Performance of domain extension on a mobile device

Benchmark	N	M	Outsource (ms)	Audit (ms)	Prove (ms)	Verify (ms)	Storage	Comm.
#1	1 K	24	8.40	0.04	0.16	1.77	96 B	32 bits
#2	1 K	2150	447.29	0.10	4.04	3.62	8.4 KB	32 bits
#3	1 K	25500	4825.01	0.14	4.09	2.31	99.6 KB	32 bits
#3	1 M	25	5318.33	0.14	91.45	925.69	100 B	32 bits
#4	1 M	121	23140.73	0.10	476.08	974.82	484 B	32 bits
#5	1 M	423	86819.68	0.13	1687.04	943.90	1.7 KB	32 bits

Table 5 Performance comparison of our protocol with existing protocols

Protocol	User computation	User storage	User comm.	Cloud computation	Cloud storage	Cloud comm.	Unit space
Our protocol	$1\text{PRG} + m\text{PRF}$	$O(\lambda)$	$O(l)$	–	$O(\|F\| + m\lambda)$	$O(n+\lambda)$	$\mathbb{F}_{2^8}$ (8 bits)
Domain extension	$1\text{PRG} + m\text{PRF}$	$O(\lambda)$	$O(l)$	–	$O(\|F\| + rm\lambda)$	$O(n+r\lambda)$	$\mathbb{F}_{2^8}$ ($8r$ bits)
Chen et al. [24]	$m \cdot (n+2)\text{Exp}$	$O(\lambda)$	$O(l)$	$(l+m+n+1)\text{Exp}$	$O(\|F\| + m\lambda)$	$O(n+\lambda)$	$\mathbb{Z}_e$ (8 bits)
Xu and Chang [10]	$m \cdot n\text{Exp} + m\text{PRF}$	$O(\lambda)$	$O(l)$	$(n-1)\text{Exp}$	$O(\|F\| + m\lambda)$	$O(1+\lambda)$	$\mathbb{Z}_p$ (128 B)
Ateniese et al. [5]	$m\text{Hash} + 2m\text{Exp}$	$O(\lambda)$	$O(l)$	$(l+1)\text{Exp} + 1\text{Hash}$	$O(\|F\| + m\lambda)$	$O(1+\lambda)$	$\mathbb{Z}_N$ (128 B)
Shacham et al. [9]	$1\text{MAC} + 1\text{Enc} + m\text{PRF}$	$O(\lambda)$	$O(l)$	–	$O(\|F\| + m\lambda)$	$O(n+\lambda)$	$\mathbb{Z}_p$ (10 B)
Shacham et al. [9]	$m\text{Hash} + m(n+1)\text{Exp}$	$O(\lambda)$	$O(l)$	$l\text{Exp}$	$O(\|F\| + m\lambda)$	$O(n+\lambda)$	$\mathbb{Z}_p$ (20 B)

min and the time for verifying is 2.4 min. Compared with our implementation on a mobile device, the time for outsourcing is 2162 times, the time for proving is 461 times and the time for verifying is 632 times of our results. We remark that this scheme is efficient for public verification and can be used for many different applications, but for the resource-constrained mobile devices, our implementation is more preferable because of efficiency and flexibility.

Domain extension. Though the system with the security level $q = 256$ is quite efficient, an 8-bit authentication tag is too short to satisfy security needs in some applications. It is easy to forge an 8-bit tag for one block from 256 possibilities in total. Consequently, we design a security mechanism with domain extension. The main idea is to extend the tag size to a multiple of 8-bit for one block. In our implementation, the vector $\boldsymbol{u}$ produced by PRG can be extended to arbitrary length and b produced by PRF can be extended to at least 128 bits during outsourcing phase. Therefore, we extend vector $\boldsymbol{u}$ to matrix $\boldsymbol{U}$ and extend element b in $\mathbb{F}_q$ to vector $\boldsymbol{b}$. The parameter r denotes the rows in matrix $\boldsymbol{U}$ and the number of element in vector $\boldsymbol{b}$. Accordingly, the tag size for one data block is $8r$ bits and the possibility to forge a tag for one data block is 2^{-8r}, we can set the security parameter r according to different applications.

Take $r = 4$ and the tag size becomes 32 bits. The experiment results are shown in Table 4. We can see the time for outsourcing is about 2.4–3.3 times and verification is about 1.6–4.5 times of original results in Table 3, the time for auditing and proving is similar to original results because Audit and the $\boldsymbol{w}$ in Prove are not changed. The additional storage and communication cost is 4 times of original system because the tag size increases to 4 times of original system. Despite of a small decrease in efficiency, this method enhances the security of the system.

Comparisons with the known protocols. We compare the performances of our protocol with some existing protocols [5, 9, 10, 24] in Table 5. Due to the lack of the source code of these protocols, we only focus the theoretical performances of the protocols. Since these protocols are very different in nature, we omit many implementation details and only focus on the main influential factors with respect to file size $|F|$, block size n, total number of blocks m, total number of queries l and security parameter λ and r. Compared with the known protocols [5, 9, 10, 24], our protocol has a considerable advantage in computation cost and communication cost, because our protocol only needs 1 PRG and m PRF operations instead of heavy modular exponentiations, and the communication cost of our protocol is only 8 bits. Compared with the PRF based protocol proposed by Shacham and Waters [9], the block length adopted in [9] is very large. Therefore, it cannot be a fine-grained mechanism and not suitable for mobile networks, i.e., p should be an 80-bit or a 160-bit prime in practical situation, which is much larger than 8 bits in our system. While our scheme with domain extension supports flexibility and is more suitable for mobile networks, i.e., it can achieve different security levels when we change the value of r. Moreover, we design the scheme based on a different tool, the homomorphic MAC in [25], and we give the first efficient SCS implementation on the mobile device. We remark that they may find different applications.

5.3 Summary of features

Our SCS system achieves by far the least computation cost and the smallest overhead, moreover, the experiment results also indicate our system is very stable and efficient in practice. Many features of our system satisfy the practical needs: e.g., the system can provide a fine-grained SCS mechanism, and the user can change the size of the component in every block and the size of block size flexibly according to the need. Besides, the system utilizes domain extension to enhance the security, the user can accordingly change the tag size for one data block. Thanks to the simple operations, the functions in our SCS system are fast and efficient. To further speed up the computation, we generate one multiplication table and one addition table and store them in RAM. The former stores 2^{16} sums of pairs of elements of $\mathbb{F}$; The latter stores 2^{16} products of pairs of elements of $\mathbb{F}$. Then addition and multiplication in the finite field can be easily replaced by searching in the lookup tables. In order to enhance stability, the system uses files on the hard disk rather than puts everything into the memory, e.g., the Outsource function reads multiple blocks, then computes tags one by one and writes them into a tag file.

6 Conclusion

In this paper, we proposed an efficient SCS system for mobile networks, which achieved less computation and smaller overhead. In addition, we implemented the proposed SCS system on a mobile device and a laptop respectively, and compared it with other schemes in theory and practice. In our implementation, we developed some useful techniques to achieve greater stability, higher efficiency. Moreover, we presented an enhanced security mechanism with domain extension to satisfy different security needs in practical applications. It is a piece of interesting work in future to investigate enhanced SCS schemes with more functionality, such as dynamic data update or efficient public verifiability.

Acknowledgements This work was supported by Strategic Priority Research Program of the Chinese Academy of Sciences (Grant Nos. XDA06010703, XDA06010701), National Natural Science Foundation of China (Grant Nos. 61472416, 61272478, 61632020), Foundation of Science and Technology on Information Assurance Laboratory (Grant No. KJ-14-002), and CREST, Japan Science and Technology Agency.

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Deswarte Y, Quisquater J J, Saïdane A. Remote integrity checking—how to trust files stored on untrusted servers. In: Proceedings of Integrity and Internal Control in Information Systems VI - IFIP TC11/WG11.5 Sixth Working Conference on Integrity and Internal Control in Information Systems (IICIS), Lausanne, 2003. 1–11

2 Filho D, Barreto P. Demonstrating data possession and uncheatable data transfer. Cryptology ePrint Archive, Report 2006/150, 2006. http://eprint.iacr.org/

3 Naor M, Rothblum G N. The complexity of online memory checking. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), Pittsburgh, 2005. 573–584

4 Schwarz T, Miller E. Store, forget, and check: using algebraic signatures to check remotely administered storage. In: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, Lisboa, 2006. 12

5 Ateniese G, Burns R C, Curtmola R, et al. Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, 2007. 598–609

6 Zhu Y, Wang H X, Hu Z X, et al. Efficient provable data possession for hybrid clouds. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, Chicago, 2010. 756–758

7 Erway C, Küpçü A, Papamanthou C, et al. Dynamic provable data possession. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, Chicago, 2009. 213–222

8 Juels A, Kaliski B. Pors: proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, 2007. 584–597

9 Shacham H, Waters B. Compact proofs of retrievability. In: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, Australia, 2008. 90–107

10 Xu J, Chang E. Towards efficient proofs of retrievability. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, Korea, 2012. 79–80

11 Ateniese G, Kamara S, Katz J. Proofs of storage from homomorphic identification protocols. In: Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2009), Tokyo, 2009. 319–333

12 Bowers K, Juels A, Oprea A. Proofs of retrievability: theory and implementation. In: Proceedings of the ACM Workshop on Cloud Computing Security, Chicago, 2009. 43–54

13 Dodis Y, Vadhan S, Wichs D. Proofs of retrievability via hardness amplification. In: Proceedings of the 6th Theory of Cryptography Conference (TCC 2009). Berlin: Springer, 2009. 109–127

14 Ateniese G, Pietro R, Mancini L, et al. Scalable and efficient provable data possession. In: Proceedings of the 4th International ICST Conference on Security and Privacy in Communication Networks (SecureComm 2008), Turkey, 2008. 1–10

15 Ma H, Zhang R. Secure cloud storage for dynamic group: how to achieve identity privacy-preserving and privilege control. In: Proceedings of the 9th International Conference Network and System Security. Berlin: Springer, 2015. 254–267

16 Wang Q, Wang C, Li J, et al. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Proceedings of the 14th European Conference on Research in Computer Security, Saint-Malo, 2009. 355–370

17 Stefanov E, Dijk M, Juels A, et al. Iris: a scalable cloud file system with efficient integrity checks. In: Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC 2012). New York: ACM, 2012. 229–238

18 Cash D, Küpçü A, Wichs D. Dynamic proofs of retrievability via oblivious RAM. In: Advances in Cryptology—EUROCRYPT 2013. Berlin: Springer, 2013. 279–295

19 Shi E, Stefanov E, Papamanthou C. Practical dynamic proofs of retrievability. In: Proceedigns of ACM Conference on Computer and Communications Security (CCS 2013), Berlin, 2013. 325–336

20 Guan C, Ren K, Zhang F, et al. Symmetric-key based proofs of retrievability supporting public verification. In: Proceedigns of the 20th European Symposium on Research in Computer Security (ESORICS 2015). Berlin: Springer, 2015. 203–223

21 Lillibridge M, Elnikety S, Birrell A, et al. A cooperative internet backup scheme. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, San Antonio, 2003. 29–41

22 Wang C, Chow S, Wang Q, et al. Privacy-preserving public auditing for secure cloud storage. IEEE Trans Comput, 2013, 62: 362–375

23 Yang K, Jia X. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans Parall Distrib Syst, 2013, 24: 1717–1726

24 Chen F, Xiang T, Yang Y, et al. Secure cloud storage meets with secure network coding. In: Proceedings of Conference on Computer Communications, Canada, 2014. 673–681

25 Agrawal S, Boneh D. Homomorphic macs: mac-based integrity for network coding. In: Proceedings of the 7th International Conference on Applied Cryptography and Network Security, Paris-Rocquencourt, 2009. 292–305

26 Cheng C, Jiang T. A novel homomorphic MAC scheme for authentication in network coding. IEEE Commun Lett, 2011, 15: 1228–1230

27 Shoup V. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/

28 Krawczyk H. Cryptographic extraction and key derivation: the HKDF scheme. In: Proceedings of the 30th Annual Cryptology Conference (CRYPTO 2010). Berlin: Springer, 2010. 631–648