J Supercomput (2019) 75:77–91
https://doi.org/10.1007/s11227-015-1556-z

A new publicly verifiable data possession on remote storage

Chun-ming Tang[1] · **Xiao-jun Zhang**[1]

Published online: 14 November 2015

Abstract In this paper, a new verifiable data possession construction supporting both private and public verifiability simultaneously is proposed from a linearly homomorphic cryptography method, which allows a server to integrate l selected block-tag pairs into a single block-tag pair as a response to user's query. In our scheme, the data owner who uses the private verification and anyone else who runs the public verification algorithm simultaneously on the same set of meta-data and based on the same setup procedure can securely authenticate the integrity of client's data file stored at cloud server without retrieving the whole original data file. Besides, in fact our simultaneous privately and publicly verifiable scheme can also be adjusted to elliptic curve group. The scheme proposed is efficient on both client and server sides, especially in computation on cloud server side, which is almost optimal among those existing publicly verifiable schemes. Here the server needs not to perform any exponent operations at all, which greatly reduces client's waiting time. Finally, we make the security

The National Natural Science Foundation of China under Grant No. 11271003, the Natural Science Foundation of Guangdong Province to Develop Major Infrastructure Projects, the Basic Research Major Projects of Department of Education of Guangdong Province under Grant No. 2014KZDXM044, the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No.20134410110003, the Project of Department of Education of Guangdong Province under Grant No 2013KJCX0146, and the Natural Science Foundation of Bureau of Education of Guangzhou under Grant No. 2012A004.

✉ Chun-ming Tang
ctang@gzhu.edu.cn

Xiao-jun Zhang
zhang4838223@gmail.com

[1] School of Mathematics and Information Science, Guangzhou University, Guangzhou 510006, People's Repuplic of China

analysis of our scheme under several cryptographic assumptions, such as difficulty of Factorization Assumption and Discrete Logarithm Problem (DLP).

Keywords Cloud · Public authentication · Verifiable data possession · Integrity

1 Introduction

With the rapid development of computer science and Internet technologies, data size is increasing geometrically, which also leads to computer users requiring larger storage space to maintain/manage their data. Under the tendency above, cloud storage is becoming a desirable choice due to its remarkable advantages, which brings massive and inexpensive storage space. Different from traditional storage, data stored in cloud storage server by users will suffer a series of security problems, such as data losing, data tampering and so on. This is all because users physically lose the absolute control over those data stored in cloud. Thus it is meaningful and necessary to devise more cryptographic methods to solve or relieve these security problems, which makes users enjoy the cloud storage service safely and assures the reputation of cloud servers to attract more clients.

Recently, there is an increasing interest in the cryptographic aspects about cloud storage, and storage in cloud may be necessary when a client's (Alice's) local storage space is too small to store her enormous data file. However, the third party (Bob), who advertises that he has the ability to store client's data file completely, may not be trustworthy. As a result, it is vital for the cautious client to ensure whether her data file is really stored there, without downloading all stored data file to validate since this may be prohibitive in terms of bandwidth and time, especially when the client executes this authentication frequently.

In some cases, the data owner is not convenient to verify the integrity of her data file stored in cloud server by herself. For example, the owner is taking a train during a journey, where she certainly can not execute the authentication alone. Consequently, she has to mandate another one to implement the verification for her. In another case, the data files stored in cloud by the owner are shared by multi authorized users. A natural and meaningful question is as to how can authorized users perform the verification for the data files. Obviously, it is not secure for the data owner to send out her private key. Therefore, it is necessary to design proof of storage schemes with public authentication in these cases.

Ateniese et al. [1] first defined publicly verifiable schemes and presented a formulation, called provable data possession, for proof of storage problem, while their publicly verifiable PDP scheme is not so efficient as our scheme in communication and computation on server's side.

Juels et al. [12] proposed the first concept about proof of retrievability (POR) and elaborated the formulation for a secure POR system. Briefly speaking, in a secure POR system, the client can extract the original data file from the information collected in polynomial number of interactions with the cloud server, if a cloud server could return a correct response that the client can accept. While compared with our solution, their first scheme (was not publicly verifiable) only supports a predefined constant number

of authentications, and their second scheme offers the unbounded public verifiability property, but the prover requires to send $\mathcal{O}(\ell)$ numbers of authenticator values and obviously it is not compact as our solution.

Shacham et al. [18] also proposed two efficient POR schemes. The first one just supports private key verification, and the second one offers the public verifiability. However, compared with our publicly verifiable data possession scheme, their publicly verifiable scenario requires a little higher computation cost on both verifier and cloud server.

Xu [21] utilized homomorphic cryptography to construct several preferable POR schemes, which only support private key verification as well. The solutions in [21] satisfy the efficiency requirement, that is, the computation cost, storage overhead, and communication cost on client are all $\mathcal{O}(\lambda)$, which is independent of the file size. Here λ is the security parameter, and our results are comparable to those scenarios in [21] on storage, communication and computation (on server side).

In [13], authors proposed the first efficient fully dynamic PDP(DPDP) solution, which supports verifying the integrity of the data file stored on cloud server, when the client updates those outsourced data files. But when it is expanded to support the public verification, it requires a little more communication and computation cost. And there are also some other solutions which extend the PDP model to support provable updates to stored data [4,5,9] including insertions at arbitrary positions, updates on existing blocks and revision control [24]. Hanser et al. proposed the first simultaneous privately and publicly verifiable (robust) PDP protocol [11], which allows the data owner to use the more efficient private verification and anyone else to run the public verification algorithm based on elliptic curves.

Recently, Yuan et al. [23] also proposed a POR scheme with public verifiability and constant communication cost based on a secure polynomial commitment scheme. Another scheme based on polynomial commitments for public verifiability has been introduced in [22]. There are also constructions for a distributed storage setting, that is, considering multiple storage severs [6,25]. In 2012, Paterson et al. treated proof-of-retrievability schemes in the model of unconditional security [16], where an adversary has unlimited computational power. In this case retrievability of the file can be modelled as error-correction in a certain code.

2 Definition of publicly verifiable data possession

In this paper, our system contains three major players: client, cloud server and verifier. The client who owns some data files intends to store them on the cloud server. The server advertises that he can completely store the client's data files. The verifier has the access to auditing the integrity of those data files without involving the client's secret knowledge.

We precisely describe the following five polynomial algorithms(KeyGen, DEncode, Challenge, GenProof, CheckProof) which compose our publicly verifiable data possession scheme:

KeyGen$(1^{\lambda}) \rightarrow (pk, sk)$ is a probabilistic key generation algorithm which is run by the client. Given a security parameter λ as input, it will return a pair of public-private key (pk, sk).

DEncode$(sk, \mathrm{F}) \rightarrow (id, \overline{F}, n, \{g_i\}_{i=1}^{n})$ is run by the client to generate the verification metadata. It takes a private key sk and a data file F as inputs, and returns an unique identifier id, an encoded file $\overline{F}$ with size n, and a tuple of public information $\{g_i\}_{i=1}^{n}$.

Challenge$(id, n) \rightarrow \mathrm{Q}$ is a probabilistic algorithm run by a client or verifier. Given a file identifier id and the file size n as input, it will return a query Q.

GenProof$(pk, id, \overline{F}, \mathrm{Q}) \rightarrow (\mathrm{M}, \sigma)$ is run by the server (prover) to generate a response for the verifier's query. It takes the public key pk, the file identifier id, the encoded file $\overline{F}$, and a challenge query Q *as input and returns a tuple of response* (M, σ).

CheckProof$(pk, \{g_i\}_{i=1}^{n}, \mathrm{Q}, (\mathrm{M}, \sigma)) \rightarrow$ accept or reject is a deterministic algorithm run by a client or verifier to validate the response generated by server. It takes a public key pk, a tuple of public information $\{g_i\}_{i=1}^{n}$, and a tuple of response (M, σ) as input and returns either accept or reject.

Definition 1 *(Publicly Verifiable Data Possession)* It is a Publicly Verifiable Data Possession(PVDP) scheme, if for any outputs returned by the algorithms (KeyGen, DEncode, Challenge, GenProof, CheckProof) defined above, the response generated by GenProof will always be accepted as a valid proof by CheckProof without involving the private key sk above.

Definition 2 *(Completeness of PVDP*[1]*)* A PVDP scheme (KeyGen, DEncode, Challenge, GenProve, CheckProof) is complete if an honest server (prover) who intactly stores client's data file and faithfully runs the algorithm GenProof to generate a response will always be accepted by the verifier.

3 Security model

Here we introduce a security game which is almost the same as the game in [21], and we adjust it to make it suitable for publicly verifiable data possession.

Setup The challenger runs the algorithms KeyGen to obtain a pair of public-private key (pk, sk). The challenger just keeps the private key sk securely and makes pk public.

Learning The adversary adaptively makes queries, where each query is one of the following:

Store query Given a data file F chosen by adversary, the challenger responds by $(id, \overline{F}, n, \{g_i\}_{i=1}^{n}) \leftarrow \mathrm{DEncode}(sk, \mathrm{F})$. That is, he (she) sends the encoded data file $\overline{F}$ together with its identifier id and the sequence of public information $\{g_i\}_{i=1}^{n}$ to the adversary. The challenger just keeps (id, n).

Verification query Given a file identifier id chosen by the adversary, if id is previously produced from store query that adversary has made, the challenger initiates the verification with adversary for the data file F dependent on the identifier id as below:

- Challenger chooses a random query Q with the meta-data n.
- Adversary generates a response R for the query Q in an arbitrary way.
- Challenger verifies the response R with the algorithm CheckProof and outputs $b \in$ {accept,reject}.

Challenger sends the decision bit b to the adversary as feedback. Otherwise, if id is not the output of any previous store query that adversary has made, then the challenger does nothing.

Committing The adversary chooses a file identifier id^* which is generated by challenger in Learning phase and commits id^* to challenger. Let F^* denote the data file associated to identifier id^*.
Retrieving The challenger initiates polynomially many PVDP verifications for the data file F^* where challenger plays the role of verifier and adversary plays the role of prover as in the Learning phase. From messages collected in these interactions with adversary, challenger extracts a data file F' using some PPT extractor algorithm. The adversary wins this game if his response makes challenger accept in the verification during the Retrieve phase for the query made by challenger. The challenger wins this game if the extracted file blocks are identical to the original one.

According to the security game, we introduce the following definition:

Definition 3 A PVDP scheme is sound, if the probability that both adversary and challenger win the security game with the different response is negligible. That is, when the adversary gives out the response (M', σ') which makes the challenger accept for the query Q generated by challenger, while the correct response is (M, σ) which is generated by the algorithm GenProof, the probability $\Pr[(M', \sigma') \neq (M, \sigma)]$ is negligible.

4 Our PVDP scheme

We describe the PVDP scheme in detail as follows:
KeyGen$(1^\lambda) \rightarrow (pk, sk)$

1. The client chooses randomly a λ bits RSA modulus $N = pq$ such that all of $p, q, p' = \frac{p-1}{2}, q' = \frac{q-1}{2}$ are primes and p, q are of the same bit lengths.
2. Sets $\emptyset = (p-1)(q-1) = 4p'q'$ and lets $\mathscr{QR}_N$ denote the subgroup of quadratic residues modulo N, which is also a cyclic group.
3. Chooses randomly a generator g of the subgroup $\mathscr{QR}_N$.
4. Chooses randomly $\tau \leftarrow Z_{\emptyset(N)}$.
5. Chooses randomly a seed denoted as seed from the key space of the pseudorandom function family $\{PRF_{seed} : \{0,1\}^{2\lambda} \rightarrow Z_{\emptyset(N)}\}$.

lets $g_\tau = g^\tau$. The public key is $pk = (N, g, g_\tau)$ and the private key is $sk = (p, q, \tau, seed)$.
DEncode(sk,F) $\rightarrow (id, \overline{F}, n, \{g_i\}_{i=1}^n)$

1. The client sets $\rho \in (0, 1)$ be a system parameter. Apply rate-ρ error erasure code on data file F to generate block $(F_0, ..., F_{n-1})$, such that each block $F_i \in \{0, 1\}^{m\lambda}$ and any ρn number of blocks F_i can recover the original file F.
2. Chooses an unique identifier $id \in \{0, 1\}^{\lambda}$ for the file F.
3. For each data block $F_i, i \in [0, n-1]$, computes an authentication tag $\sigma_i = \tau F_i + PRF_{seed}(id \parallel i) \bmod \emptyset(N)$.
4. The encoded file is $\overline{F} = \{(i, F_i, \sigma_i) : i \in [0, n-1]\}$, sends $(id, \overline{F})$ to the cloud storage server.
5. For each σ_i, computes the public information $g_i = g^{PRF_{seed}(id\|i)}, i \in [0, n-1]$ when id is given.

The encoded file is $\overline{F} = \{(i, F_i, \sigma_i) : i \in [0, n-1]\}$, the client sends $(id, \overline{F})$ to the server and just keeps (id, n) in local storage and makes $\{g_i\}_{i=1}^{n}$ public.

Note that the client dose not need to store the tuple $\{g_i\}$, and he can obtain those values by table lookup when needed. Besides, the tuple $\{g_i\}$ can be reused and the size $|\{g_i\}| = |\mathscr{QR}_N|$ due to the element g is the generator of $\mathscr{QR}_N$.

Challenge$(id, n) \rightarrow$ Q

1. A verifier chooses randomly a subset C $\subset [0, n-1]$ with size $|\ \mathrm{C}\ | = \ell$.
2. Chooses randomly weight v_i from $Z^*_{\emptyset(N)}$ for each $i \in$ C.

Sets Q=$\{(i, v_i) : i \in \mathrm{C}\}$,, and sends it to the cloud server.

GenProof(id, $\overline{F}$, Q) $\rightarrow$ (M,σ)

1. Server receives the query (id, Q) from the verifier.
2. Finds the encoded file $\overline{F}$=$\{(i, F_i, \sigma_i) : i \in [0, n-1]\}$ according to the identifier id.
3. Computes a message and a MAC value σ as follows:

$$\mathrm{M} = \sum\nolimits_{i \in C} v_i F_i \ mod N, \quad \sigma = \sum\nolimits_{i \in C} v_i \sigma_i \ mod N.$$

Cloud server sends (M,σ) to the verifier. It outputs (M, σ).

CheckProof(pk, $\{g_i\}$, Q, (M, σ)) $\rightarrow$ accept or reject

– With the public key pk, and the tuple of public information $\{g_i\}$, the verifier could check whether the following equation holds:

$$g^{\sigma} = \prod\nolimits_{i \in C} (g_i)^{v_i} g_{\tau}^{M} mod N.$$

It will output accept if the authentication succeeds; otherwise, outputs reject.

4.1 Completeness

Lemma 1 (Completeness) *The above PVDP scheme is complete under Definition 2.*

Proof The RHS of the verification equation above is

$$\begin{aligned} g^{\sigma} &= g^{\sum_{i \in C} v_i \sigma_i} mod N = g^{\sum_{i \in C} v_i PRF_{seed}(id\|i) + \sum_{i \in C} v_i \tau F_i} mod N \\ &= \prod_{i \in C} (g^{PRF_{seed}(id\|i)})^{v_i} . g_{\tau}^{M} mod N = \prod_{i \in C} (g_i)^{v_i} g_{\tau}^{M} mod N \end{aligned}$$

Thus the lemma is completely proved. □

4.2 Security

Before we analyse the security of our PVDP scheme, we introduce the following assumptions first.

The Knowledge of Exponent Assumption (KEA) is proposed in [7] and subsequently appears in many works [2,3,8,10,14]. The below is a variant version of KEA in the RSA ring given by Ateniese et al. [1].

Assumption 1 *(Knowledge of exponent assumption)* Let $N = pq$ be a RSA modulus, $g \in Z_N^*$ and s be a positive integer. For any PPT algorithm $\mathscr{A}$ that takes (N, g, g^s) as input and r as random coin and returns (C, Y) such that $Y = C^s mod N$, there exists a PPT extractor algorithm $\overline{\mathscr{A}}$ which, given (N, g, g^s, r) as input, outputs x such that $g^x = C mod N$.

Assumption 2 *(Factorization assumption* [17] We say an integer N is a RSA modulus, if $N = pq$ and all of $\frac{p-1}{2}$, $\frac{q-1}{2}$ are prime numbers and bit-lengths of p and q are equal. Then for any PPT adversary $\mathscr{A}$, the probability that $\mathscr{A}$ can factorize a randomly chosen λ bits RSA modulus, is negligible in λ.

Under the definition 3 and the security model introduced in section 4, we give the following result:

Theorem 1 *If the pseudorandom function family PRF in our scheme is secure, the discrete logarithm problem and the factorization of large integer are difficult to solve, then the PVDP scheme is sound.*

Before proving the above theorem, we introduce the following lemma first to show that any PPT adversaries who perform our PVDP scheme faithfully could obtain the secret τ with negligible probability.

Lemma 2 *If the pseudorandom function family PRF is secure, the discrete logarithm problem and the factorization of large integer are difficult to solve, then the proof (M, σ) in our PVDP scheme is unforgeable in the no-feedback setting, where all acceptance or rejection decisions are kept secure from the adversary in the security game.*

Game 1 The first is identical to the security game, except that

- all acceptance or rejection decisions are kept secure from the adversary $\mathscr{A}$. That means the challenger in the security game does not answer verification queries made by the adversary.
- Adversary $\mathscr{A}$ wins in Game 1, if $\mathscr{A}$'s forgery proof (M', σ') is accepted and it is different from the genuine proof (M, σ) generated by running GenProof honestly.

Game 2 The second game is the same as Game 1, except that the pseudorandom function PRF is replaced by a simulator PRF^{Sim}, which outputs true randomness over the range of PRF. Precisely, the function PRF^{Sim} is evaluated in the following way:

- The challenger keeps a table, which is empty at the very beginning, to store all previous encountered input-output pairs $(v; PRF^{Sim}(v))$.

– Given an input v, the challenger looks up the table for it. If there exists an entry $(v; u)$, then return u as output. Otherwise, choose u uniformly randomly from the range of PRF, insert $(v; PRF^{Sim}(v) = u)$ into the table and returns u as output.

Obviously, we have the following result:

Claim 1 If there is a non-negligible difference in an PPT adversary $\mathscr{A}$'s success probability between Game 1 and Game 2, then there exists another PPT adversary $\mathscr{B}$ which can break the security of the pseudorandom function PRF.

Claim 2 For any computationally PPT adversary $\mathscr{A}$, the probability that $\mathscr{A}$ finds the secret value τ after interacting in Game 2 is $\frac{1}{\emptyset(N)} \leq \frac{1}{p-1}$.

Proof Due to that the pseudorandom function PRF is secure, no PPT adversary could distinguish between the outputs of PRF and the real random numbers in $Z_{\emptyset(N)}$ in the security game. As a result, the secret value τ is disguised computationally in σ_i. Besides, no PPT adversaries could learn something useful about the value τ from the pk g_τ, since the DLP is difficult to solve. So we remark that no PPT adversary could learn the useful information about τ in security game. Recall that τ is chosen uniformly randomly from $Z^*_{\emptyset(N)}$. Thus the result is proved. □

The secret value τ is only involved in the tag σ_i and the public key g_τ. The secret value τ is chosen at random from group $Z_{\mathscr{O}(\mathscr{N})}$.

Claim 3 The probability that the adversary wins game 2 with the forgery proof is less than $\frac{1}{p-1}$.

Proof Suppose the adversary plays the role of cloud storage server in game 2 and wins the game with forgery proof (M′, σ'). That is,the adversary can generate the valid response (M', σ') in any way, which makes the challenger accept for his own query Q, while the correct response is (M, σ) which is generated according to the algorithm GenProof, then the verification equation holds for both the valid response (M', σ') and the correct response (M, σ). So we have

$$g^{\sigma} = \prod_{i \in C} (g_i)^{v_i} g_\tau^{M} mod N. \tag{1}$$

$$g^{\sigma'} = \prod_{i \in C} (g_i)^{v_i} g_\tau^{M'} mod N. \tag{2}$$

Dividing Eq. (1) with Equation (2) we have

$$\frac{g^{\sigma}}{g^{\sigma'}} = \frac{\prod_{i \in C} (g_i)^{v_i} g_\tau^{M'}}{\prod_{i \in C} (g_i)^{v_i} g_\tau^{M'}} mod N = g^{\sigma - \sigma'} mod N = g_\tau^{M - M'} mod N = g^{(M - M')\tau} mod N$$

Through the computing above, the adversary can obtain the following equation:

$$g^{\sigma - \sigma'} = g^{(M - M')\tau} mod N \tag{3}$$

The PPT adversary can compute $\tau = \frac{\sigma - \sigma'}{M = M'}$.

By the Claim 2, we have the following result: Pr[$\mathscr{A}$ wins game 2 with forgery proof] $\leq$ Pr[$\mathscr{A}$ finds τ in game 2] $\leq \frac{1}{p-1}$. Thus Claim 3 is proved. □

Therefore, Lemma 2 can be proved by combining Claim 1 and Claim 3. Subsequently, we prove the security in feedback setting.

Lemma 3 *If the pseudorandom function family PRF is secure, the discrete logarithm problem and the factorization of large integer are difficult to solve, then the proof (M, σ) in our PVDP scheme is unforgeable in the feedback setting, where all acceptance or rejection decisions are returned to the adversary in the security game.*

Game 1 The first game is just the same as security game; in the no-feedback setting, all acceptance or rejection decisions are kept secret from the adversary. It is identical to the Game 1 in the proof of Lemma 2.

Game 2·k This game is the same as Game 1, except that, $\mathscr{A}$ adaptively makes k verification queries in the Learning phase and all acceptance or rejection decisions are provided to $\mathscr{A}$ at the end of each query.

We describe two different verification strategies as below, where the first one is adopted by the challenger of the security game Game 2·k and the second one serves as the reference:

- *Simulated verifier* The challenger keeps a local copy of messages and MACs, where messages are chosen by the adversary in a SignQuery and MAC values are generated by the challenger in response to that SignQuery. The challenger also plays the role of an honest user. For each tuple $(M', \sigma', (i, v_i) : i \in C)$ generated by the adversary $mathcal A$ in a VerifyQuery, the challenger computes the corresponding genuine tuple $(M, \sigma, (i, v_i) : i \in C)$ from the challenger's local copy of messages and MACs. If adversary's output are the same as the genuine output, then outputs accept; otherwise outputs reject.
- *Imaginary verifier* An imaginary verification oracle $\mathscr{O}$ which somehow has access to the private keys sk.

We denote accept with the bit 1 and reject with the bit 0, and denote with $a_i \in \{0, 1\}$ the decision bit output by the imaginary verification oracle $\mathscr{O}$ for the i-th VerifyQuery made by the adversary $\mathscr{A}$ in Game 2·k; $b_i \in \{0, 1\}$ is the corresponding decision bit output by the simulated verifier. Further more, let $A_k = a_1 a_2 ... a_k \in \{0, 1\}^k$ and $B_k = b_1 b_2 ... b_k \in \{0, 1\}^k$. Thus we have

- $a_{k+1} \neq b_{k+1}$ implies the event that the adversary wins the Game 2·k. $(a_{k+1} = 1, b_{k+1} = 0)$ indicates the event that the adversary wins Game 2·k, since the adversary's output is valid (accepted by ImaginaryVerifier), but different from the genuine output (rejected by SimulatedVerifier). $(a_{k+1} = 0, b_{k+1} = 1)$ is impossible, since the PVDP scheme is correct such that all genuine MAC values are valid.
- $a_{k+1} = b_{k+1}$ indicates the event that the adversary loses Game 2·k.

Due to our proof in this phase being exactly identical to that in [21], here we just present our results and save the details.

Claim 4 Let ξ be a negligible function, such that for any PPT adversary $\mathscr{A}$, Pr[$\mathscr{A}$ wins Game 1 with the forgeable proof]$\leq \xi$. Then Pr[$\mathscr{A}$ wins Game 2·k with the forgeable proof]$\leq \xi$. If Pr[$A_k = B_k$] $\geq$ X, then Pr[$A_{k+1} = B_{k+1}$] $\geq X(1-\xi)$.

Claim 5 If Pr[$A_k = B_k$] $\geq$ X, then Pr[$A_{k+1} = B_{k+1}$] $\geq X(1-\xi)$.

Claim 6 Pr[$A_k = B_k$] $\geq (1-\xi)^k$.

Claim 7 Pr[$\mathscr{A}$ wins Game 2·k with the forgeable proof]$\leq \xi + (1-(1-\xi)^k) = (k+1)\xi + o(\xi)$.

Notice that Pr[$\mathscr{A}$ wins Game 2·k with the forgeable proof $|A_k = B_k$] $\leq$ Pr[$\mathscr{A}$ wins Game 1 with the forgeable proof], since in Game 2·k, $A_k = B_k$ indicates that the adversary gains absolutely no information from the k VerifyQuery in the Learning phase.

Therefore, Lemma 3 is concluded from Claim 7.

As we show that the response or proof (M, σ) in our PVDP scheme is unforgeable, now we start to prove our main result theorem 1 in this section.

Proof Since for random value$\tau \in Z_{\emptyset(N)}$, $\mathscr{A}$ can compute (g_τ^M, g^M) correctly. Given input $(g^\tau, (g_\tau)^s)$, the adversary $\mathscr{A}$ can output $(g^{M\tau}, (g^{M\tau})^s)$. By Knowledge of Exponent Assumption (KEA[11]), there exists a PPT extractor that can find M', such that $g^{M\tau} = g^{M'\tau} mod N$. □

Case 1 $M \neq M'$. If the two values M and M' are different, then the difference M-M' has to be a multiple of $\frac{1}{4}\emptyset(N)$, with which the factorization of N can be completed by Miller's result [15]. As a result, this case occurs with negligible probability under large integer factorization assumption.

Case 2 $M = M'$. Certainly the challenger wins the security game when the equation $M = M'$ holds. Here the value of M' is faithfully computed as $M' = \sum_{i \in C} v_i F_i$, which is a linear equation for the challenger with the set of weight $\{v_i\}_{i \in C}$. As a consequence, to obtain $\ell = |C|$ numbers of independent linear equations about the unknown quantities $F_i, i \in C$, the challenger can execute the protocol $\ell = |C|$ times for the same index set C. So the original file blocks $F_i, i \in C$ can be recovered by solving the linear equation system.

Through the above analysis, we obtain the following claim:

Claim 8 The case where adversary wins the security game including the above cases means that either case 1 or case 2 occurs. That is

$$\text{Pr[adversary wins the security game]} = \text{Pr[case 1 occurs]} + \text{Pr[case 2 occurs]},$$

where Pr[case 1 occurs] is negligible, and case 2 occurs means the challenger wins the security game.

With the definition 3, we proved theorem 1 completely.

4.3 Performance analysis

Due to the homomorphism, the server integrates the block-MAC pairs (F_i, σ_i) inquired by the verifier into a single block-MAC pair (M, σ) as a response returned to the verifier, which makes our scheme efficient in communication and computation (on server side): $\mathcal{O}(\lambda)$ is the communication cost and $\mathcal{O}(\lambda)$ is the storage overhead on both client and server side, where λ is the bit-length of the RSA modulus N. For each query initiated by a verifier, the size of response returned by server is 2λ bits. And the server only needs to perform 2ℓ mul and 2ℓ add operations to generate such a response, which is optimal considering those existing publicly verifiable schemes. On receiving the server's response, the verifier requires to perform $\ell + 2$ exp and $\ell + 1$ mul operations to conduct the verification, which is also comparable to those existing verifiable schemes. As a result, all of these evaluation overhead grows linearly to the number of elements in the query. Due to the tag and $\sigma_i \in \{0, 1\}^{\lambda}$ and $F_i \in \{0, 1\}^{m\lambda}$, the storage overhead is $|F|(1 + \frac{1}{m})$. However, in the setup, one group multiplication, one group addition and one PRF evaluation are required to compute a tag per each data block. Besides, a group exponentiation is also required to generate the public information $\{g_i\}$, and all of the preprocessing can be conducted off-line by the client. Here $\ell = |C|$ is the number of indices selected during a verification.

We now compare our PVDP scheme with some existing schemes [1,11,18–20,22,23] in terms of communication cost, computation cost on server side, computation cost on verifier side and storage cost. For simplicity, here we denote them as comm.cost, comp.cost(S), comp.cost(V), storage.cost, respectively, and publicly+privately verifiability presents the simultaneously public and private verifiability. In Table 1, λ is the security parameter. $|G|$ denotes the size of group/field and m represents the size of a data block ($m = 100$ is a good choice). In each scheme, ℓ denotes the size of a challenge or query set $(i, v_i) : i \in C$ and we denote multiplication, exponentiation and addition in the corresponding groups (fields) as mul, exp and add, respectively. Here we ignore all the hash functions, pseudorandom functions and pairing operations, since all of these operations are not required during the verification phase in our scheme, which makes our PVDP scheme more efficient when users perform the verification many times.

Here we just focus on the size of response/proof for the communication cost, because it is at the mercy of the response/proof size. Note that the client/verifier dose not need to store the tuple $\{g_i\}$, and he can obtain those values by table lookup when needed(the table can be stored at his facebook or even at the cloud server with a secure signature as in [18]). Besides, the tuple $\{g_i\}$ can be reused and the size $|\{g_i\}| = |\mathcal{QR}_N|$ due to the element g is the generator of $\mathcal{QR}_N$.

In our PVDP scheme, the expensive cost for the public information $\{g_i\}$ just happens in setup phase (algorithm DEncode), which is just the same as what most previous works did when they evaluated the authentication tags (values) [11,19,20,22,23]; thus we just focus the verifier's computation cost on the verification phase as art protocols did.

Table 1 Complexity comparision

Scheme	Publicly+privately verifiability	Communication.cost	Storage.cost	Computation.cost(S)	Computation.cost(V)
[2]	No	2λ	$\mathcal{O}(\lambda)$	$2\ell\ mul + \ell\ exp + \ell\ add$	$(\ell+2)\ exp + (\ell+1)\ mul$
[4]	No	$m\lambda + \lvert G\rvert$	$\mathcal{O}(\lambda)$	$(1+m)\ell mul + \ell\ exp + m\ell\ add$	$(m+\ell)exp + (\ell+m)\ mul$
[6]	No	2λ	$\mathcal{O}(\lambda)$	$2\ell\ mul + \ell exp + \ell\ add$	$(\ell+1)\ exp + (\ell+1)mul$
[8]	No	$\lambda + 2\lvert G\rvert$	$\mathcal{O}(\lambda)$	$(2\ell+2)\ mul + \ell exp + (\ell+1)\ add$	$(3+\ell)\ exp + (\ell+2)mul$
[9]	No	2λ	$\mathcal{O}(\lambda)$	$2\ell\ mul + 2\ exp + 2\ell\ add$	$3\ exp + (1+\ell)\ mul + \ell\ add$
[10]	No	$\lambda + 2\lvert G\rvert$	$\mathcal{O}(\lambda)$	$(\ell+m)\ mul + (\ell+m)\ exp + \ell\ add$	$(2+\ell)\ exp + 2\ell\ mul$
[25]	Yes	$(m+1)\lambda$	$\mathcal{O}(\lambda)$	$(2\ell+1)\ mul + (\ell+m)\ exp + 2\ell\ add$	$(m+\ell)\ exp + 2\ell\ mul$
Our PVDP	Yes	2λ	$\mathcal{O}(\lambda)$	$\ell\ mul + \ell\ add$	$(2+\ell)exp + (\ell+1)\ mul$

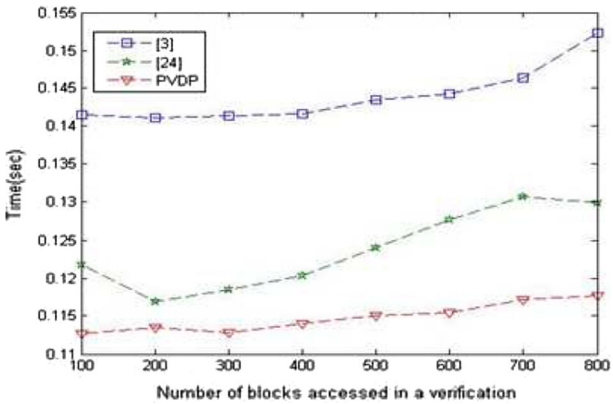

Fig. 1 Comparison of three schemes

4.4 Experimental results

We have modelled a prototype of our PVDP scheme as well as the schemes [18,20] to show the efficiency of our PVDP scheme by the MATLAB with version 7.12.0.635 (R2011a).Our implementation is not optimized and further performance improvements of our scheme can be expected.

Our text machine is a laptop computer, which is equipped with a 2GHz AMD Athlon(tm) X2 Dual-Core, a 1GB RAM. All experimental results are the mean of 10 trials. We denote the test data file as the array where the elements means the data block size. Since all experiment results vary little across different trials, we do not focus on the variances.

To illustrate the feature of computational time for the verification, we vary the number of data blocks accessed in the verification from 100 to 800. Our experiment results are shown in Fig. 1, which indicates that the authenticating time is proportional to the number of blocks accessed in a verification, which is also at the mercy of the challenge. Note that the size of the challenge is bounded. Thus the result of the experiment is just consistent with our analysis.

5 Conclusion

In this paper, we present a new verifiable data possession (PVDP), which allows both private and public verifiability simultaneously to verify the integrity of clients data files stored on cloud server without downloading all of those files. Sequently, we prove the security of our scheme. Our solution is efficient in communication, storage, and computation on servers side. Especially, the computation cost on server side is preferable to those most existing publicly verifiable schemes without increasing the verifier's workload, which greatly reduces client's waiting time. However, our work also encounters same dilemma as previous researches, that is, the client requires large computation cost in setup phase (although the client can conduct this off-line), which is the main drawback of our solution. In future research, we will devise preferable

publicly verifiable data possession scheme which supports more features, and the computation cost is far more efficient as well on the verifier side.

References

1. Ateniese G, Burns R, Curtmola R et al (2007) Provable data possession at untrusted stores. In: CCS '07: ACM conference on computer and communications security, pp 598–606
2. Bellare M, Palacio A (2004) The knowledge-of exponent assumptions and 3-round zero knowledge protocols. In: Advances in CRYPTO '04, pp 273–289
3. Bellare M, Palacio A (2004) Towards plaintext aware public-key encryption without random oracles. In: ASIACRYPT '04: Inter-national conference on the theory and application of cryptology and information Security, pp 48–62
4. Cash D, Kp A, Wichs D (2013) Dynamic proofs of retrievability via oblivious ram. Advances in cryptologyCEUROCRYPT 2013. Springer, Berlin Heidelberg, pp 279–295
5. Chen B, Curtmola R (2012) Robust dynamic provable data possession. In: ICDCS Workshops, pp 515–525
6. Curtmola R, Khan O, Burns RC, Ateniese G (2008) Mr-pdp: multiple-replica provable data possession. In: ICDCS 2008:411–420
7. Damgard I (1992) Towards practical public key systems secure against chosen ciphertext attacks. In: Advances in CRYPTO '91, pp 445–456
8. Dent AW (2006) The cramer shoup encryption scheme is plaintext aware in the standard model. In: advances in EUROCRYPT '06, pp 289–307
9. Erway C, Kp A, Papamanthou C et. al. (2009) Dynamic provable data possession. In: Proceedings of the 16th ACM conference on computer and communications security, pp 213–222
10. Hada S, Tanaka T (1998) On the existence of 3-round zero knowledge protocols. In: Advances in CRYPTO '98, pp 408–423
11. Hanser C, Slamanig D (2013) Efficient simultaneous privately and publicly verifiable. In: SECRYPT'13. http://eprint.iacr.org/2013/392
12. Juels A, Kaliski B (2007) Pors: proofs of retrievability for large files. In: CCS '07: ACM conference on computer and communications security, pp 584-597
13. Kp A (2010) Efficient cryptography for the next generation secure cloud. A adviser-Lysyanskaya, pp 104–149
14. Krawczyk H (2005) HMQV: a high performance secure diffie Hellman protocol. In: Advances in CRYPTO '05, pp 546–566
15. Miller G (1975) Riemann's hypothesis and tests for primality. In: STOC'75: ACM symposium on Theory of Computing, pp 234–239
16. Paterson MB, Stinson DR, Jalaj Upadhyay (2012) A coding theory foundation for the analysis of general unconditionally secure proof-of-retrievability schemes for cloud storage. http://eprint.iacr.org/2012/611
17. Rivest R, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126
18. Shacham H, Waters B (2008) Compact Proofs of Retrievability. In: ASIACRYPT '08: International conference on the theory and application of cryptology and information security, pp 90–107
19. Wang C, Chow S, Wang Q et al (2011) Privacy preserving public auditing for secure cloud storage. IEEE Comp Soc 62(2):362–375
20. Wang Q, Wang C, Li J et al (2009) Enabling public verifiability and data dynamics for storage security in cloud computing. In: ESORICS'09: European conference on Research in computer security, pp 355–370
21. Xu J (2012) Towards efficient proofs of storage and verifiable outsourced database in cloud computing. http://scholarbank.nus.edu.sg/bitstream/handle/10635/33347/xujia-thesis-A0002244B-May15-2012?sequence=1
22. Xu J, Chang E (2012) Towards efficient proofs of retrievability. In: proceedings of AsiaCCS '12, pp 79–80
23. Yuan J, Yu S (2013) Proofs of retrievability with public verifiability and constant communication cost in cloud. In: Proceedings of Asia CCS-SCC '13, pp 19–26

24. Zhang Y, Blanton M (2013) Efficient dynamic provable possession of remote data via balanced update trees. In: Proceedings of AsiaCCS, pp 183–194
25. Zhu Y, Hu H, Ahn GJ, Yu M (2012) Cooperative provable data possession for integrity verification in multicloud storage. IEEE Trans Parallel Distrib Syst 23(12):2231–2244