Identity-Based Remote Data Possession Checking in Public Clouds

Huaqun Wang, Qianhong Wu, Bo Qin, and Josep Domingo-Ferrer

Universitat Rovira i Virgili

Department of Computer Engineering and Mathematics

Av. Països Catalans 26, E-43007 Tarragona, Catalonia

E-mail {huaqun.wang, qianhong.wu, bo.qin, josep.domingo}@urv.cat

Abstract

Checking remote data possession is of crucial importance in public cloud storage. It enables the users to check that their outsourced data have been kept intact without downloading the original data. The existing remote data possession checking (RDPC) protocols have been designed in the PKI (public key infrastructure) setting. The cloud server has to validate the users' certificates before storing the data uploaded by the users in order to prevent spam. This incurs considerable costs since numerous users may frequently upload data to the cloud server. This paper addresses this problem with a new model of identity-based RDPC (ID-RDPC) protocols. We present the first ID-RDPC protocol proven to be secure assuming the hardness of the standard computational Diffie-Hellman (CDH) problem. In addition to the structural advantage of elimination of certificate management and verification, our ID-RDPC protocol also outperforms existing RDPC protocols in the PKI setting in terms of computation and communication.

Index Terms

Remote data possession checking, identity-based cryptography, cloud computing.

1 Introduction

Advances in networking and computing technologies have prompted many organizations to outsource their storage needs on demand. This new economic and computing paradigm is commonly referred to as cloud storage. It brings appealing benefits including relief of the burden for storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc. However, there are barriers that hinder migration to the cloud. One of the main barriers is that, due to lack of physical control over the outsourced data, a cloud user may worry about whether her data are kept as expected. If the cloud user is a company, apart from the risk of remote malicious attacks on the cloud, the traditional concerns posed by malicious company insiders are now supplemented by the even more hazardous threat of malicious outsiders who are given the power of insiders. A recent EU bill forces companies migrating to the cloud to be liable for any data corruption or privacy breach into which their cloud service provider (CSP) may incur, even when they do not retain control over their data. Convincing cloud users that their data are intact is especially vital when users are companies. Remote data possession checking (RDPC) is a primitive designed to address this issue.

1.1 Related work

RDPC allows a client that has stored data at a public cloud server (PCS) to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. In order to achieve secure RDPC implementations, Ateniese *et al.* proposed a provable data possession (PDP) paradigm [1] and designed two provably-secure PDP schemes based on the difficulty of large integer factoring. They refined the original paradigm and proposed a dynamic PDP scheme in [2] but their proposal does not support the insert operation. In order to solve this problem, Erway *et al.* proposed a full-dynamic PDP scheme by employing an authenticated flip table [3].

Following Ateniese *et al.*'s pioneering work, researchers devoted great efforts to RDPC with extended models and new protocols [4], [5], [6], [7], [8], [9], [10], [11], [12]. One of the variations is the proof of retrievability (POR), in which a data storage server cannot only prove to a verifier that he is actually storing all of a client's data, but also it can prove that the users can retrieve them at any time. This is stronger than the regular PDP notion. Shacham presented the first POR schemes [15] with provable security. The state of the art can be found in [16], [17], [18], [19] but few POR protocols are more efficient than their PDP counterparts. The challenge is to build POR systems that are both efficient and provably secure [14]. Note that one of benefits of cloud storage is to enable universal data access with independent geographical locations. This implies that the end devices may be mobile and limited in computation and storage. Regular RDPC protocols are more suitable for cloud users equipped with mobile end devices. Our ID-RDPC architecture and protocol are based on the PDP model.

1.2 Motivation and contribution

This paper focuses on RDPC in company-oriented cloud storage. Consider a scenario in which a company purchases the cloud storage service. Only the staff members of the company are allowed to upload data to the PCS and may check the integrity of their data with mobile devices. PCS has to be convinced that the data (and their tags) to be uploaded come from the staff of the company, although this step is usually omitted in the existing models. The metadata or the tags are indeed signatures of the original data. Note that the existing RDPC protocols are designed in the PKI setting. PCS needs to validate the tags and the appended public key certificate of the users. The validation of the legit uploads incurs considerable overheads since the staff may frequently upload data to PCS. This burden can only be *partially* mitigated by letting PCS cache the verified certificates. Indeed, caching cannot be used for certificates revoked before their expiration, for employees who leave the company, for newly recruited employees, etc. In addition to the heavy certificate verification, the system suffers from complicated certificate management: certificates generation, delivery, revocation, renewal, etc.

In order to solve the above problem, we investigate a new RDPC model incorporating identity-based cryptography, *i.e.*, the ID-RDPC model. Our contribution is twofold:

- First, we formalize the ID-RDPC model. In this model, a trusted private key generator (PKG) generates the system public key and the master secret key. The PKG also generates private keys for the clients, *i.e.*, the staff members of the company, by taking as input the staff members' identities and the PKG's master secret key. With a private key, the client can generate the tags of the data to be uploaded. Upon receiving a request of a data possession proof, the PCS can generate the proof without verifying any certificate but simply checking that the corresponding system public key is from a company allowed to use the service. Finally, the client can verify whether the PCS-generated proof is valid.
- Second, we realize the first ID-RDPC protocol. The main challenge to design the ID-RDPC protocol is that it requires the client to generate aggregatable ID-based signatures like tags without applying the hash-and-sign paradigm to the original data. We address this with a variation of the well-known Schnorr signature. The instantiated ID-RDPC protocol is shown to be secure by assuming the hardness of the Computational Diffie-Hellman problem. In addition to the structural advantage of eliminating certificate management and verification, our ID-RDPC protocol is more efficient than the existing RDPC protocols in the PKI setting in terms of computation and communication.

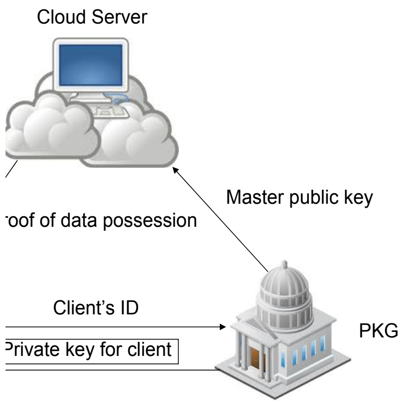

Fig. 1. The System Model of ID-RDPC

1.3 Paper Organization

The rest of this paper is organized as follows. Section 2 formalizes the ID-RDPC model. Section 3 presents our ID-RDPC protocol. Section 4 evaluates the protocol performance. Section 5 analyzes the protocol security. Finally, Section 6 concludes the paper.

2 Modeling ID-RDPC

The ID-RDPC system model and its security definition are given in this section. An ID-RDPC protocol comprises three different entities, as illustrated in Figure 1. They can be identified as follows:

1) *PKG (Private Key Generator)*. Entity, trusted by the clients and the PCSs, that generates the public parameters *Params*, the master public key *mpk*, the master secret key *msk* and the private key of the *Client* defined below.
2) *Client*. Entity which has massive data to be stored on the public cloud for maintenance and computation. Clients can be either individual consumers or group consumers, *e.g.*, the departments of the company in the motivated scenario.
3) *Cloud Server*. Entity, managed by the cloud service provider, that has significant storage space and computational resources to maintain the clients' data.

In the cloud paradigm, by putting the large data files on the remote cloud servers, the clients can be relieved of the burden of storage and computation. As the clients no longer possess their data locally, it is of critical importance for them to ensure that their data are being correctly stored and maintained. That is, clients should be equipped with certain security means so that they can periodically verify the correctness of the remote data even without the existence of local copies.

We next formally define an ID-RDPC scheme. We then give a security definition to capture its security requirements.

Definition 1 (ID-RDPC): An ID-RDPC protocol is a collection of five polynomial time algorithms (Setup, Extract, TagGen, GenProof, CheckProof) running as follows:

1) $(params, mpk, msk) \leftarrow \mathsf{Setup}(1^k)$ is the parameter-generation algorithm. It takes as input the security parameter k and outputs the system public parameters *params*, the master public key *mpk* and the master secret key *msk*.
2) $(pk_{ID}, sk_{ID}) \leftarrow \mathsf{Extract}(1^k, params, mpk, msk, ID)$ is a probabilistic key-extraction algorithm that is run by PKG to extract the client's private key. It takes as inputs the public parameters $params$, the master public key mpk, the master secret key msk, and the identity ID of a client. $\mathsf{Extract}(\cdot)$ outputs the private key sk_{ID} corresponding to the client with identity *ID*.
3) $T_m \leftarrow \mathsf{TagGen}(sk_{ID}, m)$ is an algorithm that is run by the client *ID* to generate the verification metadata. It takes as input a secret key sk_{ID} and a file block m, and returns the verification tag T_m.
4) $V \leftarrow \mathsf{GenProof}(params, F, chal, \Sigma, ID)$ is run by the PCS in order to generate a proof of data possession. It takes as inputs the public parameter $params$, the client's identity ID, an ordered collection F of blocks, a challenge $chal$ and an ordered collection Σ of the verification tag corresponding to the blocks in F. It returns a proof of data possession V for the blocks in F that are determined by the challenge $chal$.

5) $\{\text{“}success\text{”}, \text{“}failure\text{”}\} \leftarrow \mathsf{CheckProof}(mpk, sk_{ID}, chal, V)$ is run by the client in order to validate a proof of data possession. It takes as inputs the master public key mpk, the master secret key sk, a challenge $chal$ and a proof of data possession V. It returns “success” or “failure”, representing that V is a correct proof or not.

In the CheckProof phase, if the private key sk_{ID} is necessary, the ID-RDPC protocol is called private ID-RDPC. The ID-RDPC we propose in this paper belongs to this type.

In addition to communication and computation overheads as low as possible, an ID-RDPC protocol should satisfy the following requirements:

1) The verifier should not be required to keep an entire copy of the file(s) to be checked. It would be impractical for a verifier to replicate the content of all provers to be verified. Storing a reduced-size digest of the data at the verifier should be sufficient for verification of the PCS-generated proof.
2) The protocol has to stay secure even if the prover is malicious. A malicious prover is interested in proving knowledge of some data that she does not entirely know; security means that such a prover will fail in convincing the verifier.
3) It ought to be possible to run the verification an unlimited number of times.

To capture the above security requirements, we define the security of an ID-RDPC protocol as follows.

Definition 2 (Unforgeability): A ID-RDPC protocol is secure if for any (probabilistic polynomial) adversary $\mathcal{A}$ the probability that $\mathcal{A}$ wins the ID-RDPC game on a set of file blocks is negligible. The ID-RDPC game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ can be depicted as follows:

1) Setup: The challenger runs $(params, mpk, msk) \leftarrow KeyGen(1^k)$, sends $(params, mpk)$ to the adversary $\mathcal{A}$ and keeps confidential the master secret key msk.
2) First-Phase Queries: The adversary $\mathcal{A}$ adaptively makes a number of different queries to the challenger $\mathcal{C}$. Each query can be one of the following:

- Extract queries. The adversary can ask for the private key of any identity ID. The challenger obtains the private key sk_{ID} by running $\mathsf{Extract}(params, mpk, ID)$ and forwards sk_{ID} to the adversary. Denote the extracted identity set by S_1.
- Hash queries. The adversary makes *hash* function queries adaptively. The challenger responds with the *hash* values to the adversary.
- Tag queries. The adversary makes block-tag pair queries adaptively. For a query m received from the adversary, the challenger computes the tag $T_m \leftarrow \mathsf{TagGen}(sk_{ID}, m)$ and sends it back to the adversary. Without loss of generality, let $\{(m_i, T_i) : i \in \mathbb{I}_1\}$ be the set of queried block-tag pairs.

3) Challenge: The challenger generates a challenge $chal$ which defines a ordered collection $\{ID^*, i_1, i_2, \cdots, i_c\}$, where $ID^* \notin S_1$ is the identity of a non-corrupted client, $\{i_1, i_2, \cdots, i_c\} \nsubseteq \mathbb{I}_1$, and c is an positive integer. The adversary is required to provide a proof of data possession checking for the blocks $m_{i_1}, \cdots, m_{i_c}$.
4) Second-Phase Queries: Similar to the First-Phase Queries. Suppose that the Extract query identity set is S_2 and $\{(m_i, T_i) : i \in \mathbb{I}_2\}$ is the set of queried block-tag pairs in this second phase. The restriction is that $\{i_1, i_2, \cdots, i_c\} \nsubseteq \mathbb{I}_1 \cup \mathbb{I}_2$ and $ID^* \notin S_1 \cup S_2$.
5) Forge: The adversary $\mathcal{A}$ computes a proof of data possession checking V for the blocks indicated by $chal$ and returns V.

We say that the adversary wins in the ID-RDPC game if **CheckProof**$(pk_{ID^*}, sk_{ID^*}, chal, V)$ = "$success$".

Definition 2 states that, for the challenged blocks, an untrusted PCS cannot produce a proof of data possession if the blocks have been modified or deleted. This is not sufficient for the ID-RDPC protocol because the definition does not state clearly the status of the blocks that are not challenged. In practice, a secure RDPC protocol also needs to guarantee that after validating the PCS-generated proof, a client can be convinced that all of his outsourced data have been kept intact with a high probability. This observation gives the following security definition.

Definition 3 ((ρ, δ) security): An ID-RDPC protocol is (ρ, δ) secure if, given a fraction ρ of PCS-corrupted blocks, the probability that the corrupted blocks are detected is at least δ.

3 PROPOSED ID-RDPC PROTOCOL

In this section, we present an efficient ID-RDPC protocol. The proposal is built based on bilinear pairings, which are briefly reviewed below.

3.1 Bilinear pairings

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two cyclic multiplicative groups with the same prime order q, *i.e.*, $|\mathcal{G}_1| = |\mathcal{G}_2| = q$. Let $e : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$ be a bilinear map [22], which satisfies the following properties:

1) Bilinearity: $\forall g_1, g_2, g_3 \in \mathcal{G}_1$ and $a, b \in \mathcal{Z}_q$,

$$e(g_1, g_2g_3) = e(g_2g_3, g_1) = e(g_2, g_1)e(g_3, g_1)$$
$$e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$$

2) Non-degeneracy: $\exists g_4, g_5 \in \mathcal{G}_1$ such that $e(g_4, g_5) \neq 1_{\mathcal{G}_2}$.

3) Computability: $\forall g_6, g_7 \in \mathcal{G}_1$, there is an efficient algorithm to calculate $e(g_6, g_7)$.

Such a bilinear map e can be constructed with the modified Weil [20] or Tate pairings [21] on elliptic curves. Our ID-RDPC scheme is constructed on the gap Diffie-Hellman group, where the computational Diffie-Hellman (CDH) problem is hard while the decisional Diffie-Hellman (DDH) problem is easy.

Definition 4 (Gap Diffie-Hellman (GDH) Group): Let g be a generator of $\mathcal{G}_1$. Given $g, g^a, g^b, g^c \in \mathcal{G}_1$ and unknown $a, b, c \in \mathcal{Z}_q^*$, an efficient algorithm exists to determine whether $ab = c \bmod q$ holds by verifying $\hat{e}(g^a, g^b) = \hat{e}(g, g)^c$ in polynomial time (DDH problem), while no efficient algorithm exists to compute $g^{ab} \in \mathcal{G}_1$ with non-negligible probability within polynomial time (CDH problem). A group $\mathcal{G}_1$ is a (t, ϵ)-GDH group if the DDH problem can be efficiently solved, while no algorithm (t, ϵ)-breaks the CDH problem.

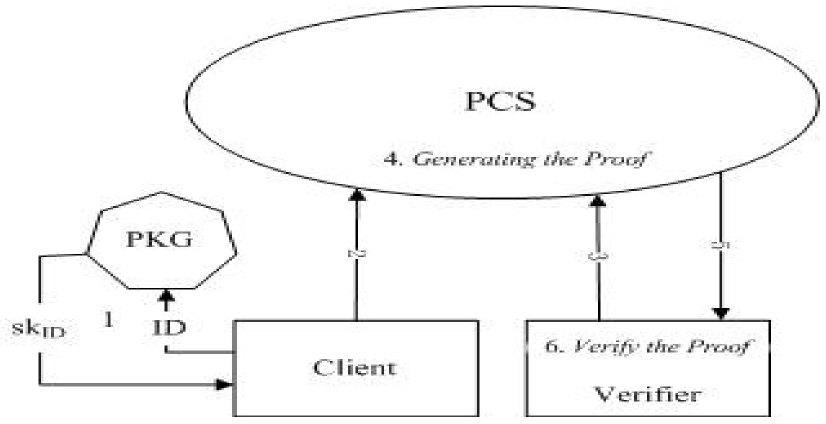

Fig. 2. ID-RDPC Architecture

3.2 ID-RDPC protocol construction

This protocol comprises the procedures Setup, Extract, TagGen, CheckTag, GenProof and CheckProof. The protocol architecture is depicted in Figure 2. The figure can be described as follows: 1. *PKG* creates the private key for the client in Extract. 2. The client generates the block-tag pairs and uploads them to PCS. 3. The verifier (who can be the client herself) sends the challenge to PCS. 4. PCS creates the possession proof. 5. PCS sends the possession proof to the verifier. 6. The verifier checks the possession proof.

Suppose that the number of stored message blocks is n. We describe below the procedures of the ID-RDPC scheme.

- Setup: PKG chooses a random number $x \in \mathcal{Z}_q^*$ and sets $Y = g^x$, where g is a generator of the group $\mathcal{G}_1$. PKG chooses a random element $u \in \mathcal{G}_1^*$. Define two cryptographic hash functions: $H : \{0,1\}^* \to \mathcal{Z}_q^*$, $h : \mathcal{Z}_q^* \to \mathcal{G}_1^*$. Let f be a pseudo-random function and let π be a pseudo-random permutation:

$$f : \mathcal{Z}_q^* \times \{1, 2, \cdots, n\} \to \mathcal{Z}_q^*$$

$$\pi : \mathcal{Z}_q^* \times \{1, 2, \cdots, n\} \to \{1, 2, \cdots, n\}$$

  PKG publishes $\{\mathcal{G}_1, \mathcal{G}_2, e, q, g, Y, u, H, h, f, \pi\}$ and keeps x as the *master key*.

- Extract. A client submits his identity ID to PKG. PKG picks $r \in \mathcal{Z}_q^*$ and computes

$$R = g^r, \qquad \sigma = r + xH(ID, R) \bmod q$$

PKG sends the private key $sk_{ID} = (R, \sigma)$ to the client by a secure channel. The client can verify the correctness of the received private key by checking whether $g^\sigma \overset{?}{=} R \cdot Y^{H(ID,R)}$ holds. If the previous equality holds, the client accepts this private key; otherwise, she rejects it.

- TagGen(sk_{ID}, F, i): We assume that the client generates the tags sequentially according to the counter i. That is, the client generates a tag for a message block m_2 after m_1, which implies that the client maintains the latest value of the counter i. For m_i, the client performs the TagGen procedure as follows:
  1) Compute $T_i = (h(i)u^{m_i})^\sigma$.
  2) Output T_i and send (m_i, T_i) to the PCS.

- GenProof$(mpk, \mathcal{F} = (m_1, m_2, \cdots, m_n), chal, \Sigma = (T_{m_1}, \cdots, T_{m_n}))$: In this phase, the verifier (who can be the client herself) queries the PCS for a proof of data possession of c file blocks whose indices are randomly chosen using a pseudo-random permutation keyed with a fresh randomly-chosen key for each challenge. The number $k_1 \in \mathcal{Z}_q^*$ is the random key of the pseudo-random permutation π. Also, $k_2 \in \mathcal{Z}_q^*$ is the random key of the pseudo-random function f. Let $chal = (c, k_1, k_2)$. Then, the PCS does:
  1) For $1 \leq j \leq c$, compute the indices and coefficients of the blocks for which the proof is generated: $i_j = \pi_{k_1}(j), a_j = f_{k_2}(j)$. In this step, the challenge $chal$ defines an ordered set $\{c, i_1, \cdots, i_c, a_1, \cdots, a_c\}$.
  2) Compute $T = \prod_{j=1}^{c} T_{i_j}^{a_j}$, $\hat{m} = \sum_{j=1}^{c} a_j m_{i_j}$.
  3) Output $V = (T, \hat{m})$ and send V to the client as the response to the $chal$ query.

- CheckProof$(mpk, sk_{ID}, chal, V)$: Upon receiving the response V from the PCS, the verifier (who can be the client herself) does:

1) Check the equation

$$e(T,g) \overset{?}{=} e(\prod_{i=1}^{c} h(\pi_{k_1}(i))^{f_{k_2}(i)} u^{\hat{m}}, R \cdot Y^{H(ID,R)}).$$

2) If the previous equation holds, output "success". Otherwise output "failure".

An ID-RDPC protocol must be workable and correct. That is, if the PKG, the Client and the PCS are honest and follow the specified procedures, the response V can pass the $CheckProof$ phase. The correctness of our ID-RDPC protocol follows from a direct verification:

$$\begin{aligned} e(T,g) &= e(\textstyle\prod_{j=1}^{c} T_{i_j}^{a_j}, g) \\ &= e(\textstyle\prod_{j=1}^{c} (h(i_j) u^{m_{i_j}})^{\sigma a_j}, g) \\ &= e(\textstyle\prod_{j=1}^{c} (h(i_j) u^{m_{i_j}})^{a_j}, g^{\sigma}) \\ &= e(\textstyle\prod_{j=1}^{c} h(i_j)^{a_j} u^{\sum_{j=1}^{c} a_j m_{i_j}}, R \cdot Y^{H(ID,R)}) \\ &= e(\textstyle\prod_{j=1}^{c} h(\pi_{k_1}(j))^{f_{k_2}(j)} u^{\hat{m}}, R \cdot Y^{H(ID,R)}) \end{aligned}$$

4 Performance Analysis

We analyze the performance of our proposed ID-RDPC protocol. Comparisons with up-to-date RDPC protocols (in the PKI setting) show that our protocol outperforms them in terms of computational and communication overheads. Note also that, unlike the existing protocols in the PKI setting, our protocol does not suffer from resource-consuming certificate management and verification.

Computation: In our ID-RDPC protocol, suppose that there exist n message blocks. In the *TagGen* phase, the client needs to do $2n$ exponentiations and n multiplications on $\mathcal{G}_1$. In the *GenProof* and *CheckProof* phases, the client needs to do 2 pairings, $c+1$ exponentiations and $c+1$ multiplications on $\mathcal{G}_1$. At the PCS's side, the cloud server needs to do c exponentiations and $c-1$ multiplications on $\mathcal{G}_1$. The exponentiation and scalar multiplication operations are more efficient than pairing and usually consume much less time [26] than the pairing map. Other operations like hashing and permutation are omitted since they just contribute little computation cost, compared with

TABLE 1

Comparison of communication overheads (bits)

Schemes	Query	Response	Storage	ID-based
[1]	$4\mathcal{Z}_N^*(4096)$	$2\mathcal{Z}_N^*(2048)$	O(1)	No
[4]	$2\mathcal{Z}_N^*(2048)$	$1\mathcal{Z}_N^*(1024)$	O(n)	No
Ours	$3\mathcal{Z}_q^*(480)$	$1\mathcal{G}_1+1\mathcal{Z}_q^*(480)$	O(1)	Yes

a bilinear map, an exponentiation or a scalar multiplication. Based on the test presented in [23], [24], [25], a Tate pairing operation consumes about 10 ms on a PIII 3.0 GHz platform, 30 ms on a Pentium D 3.0 GHz platform and 170 ms on an iMote2 sensor working at 416 MHz. The underlying field of the elliptic curve is based on a 512-bit prime k, which achieves a security level similar to a 1024-bit RSA. Our proposed RDPC protocol is more efficient than the RSA-based RDPC schemes [1], [2], [3], [4] in terms of computational overhead. This seems desirable in the motivated scenario in which the clients may be implemented in mobile devices having limited computation power.

Communication: The U.S. National Bureau of Standards and ANSI X9 have determined the shortest key length requirements: 1024 bits for RSA and DSA, 160 bits for ECC [28]. According to the standards, we analyze the communication overhead of our scheme, which mainly comes from the ID-RDPC queries and responses. In an ID-RDPC query, the client needs to send 3 elements in $\mathcal{Z}_q^*$ to PCS. In an ID-RDPC response, the PCS needs to respond with 1 element in $\mathcal{G}_1$ and 1 element in $\mathcal{Z}_q^*$ to the client. The total communication is about 3*160+160+320=960 bits. This amount of communication is perfectly affordable with current communication technologies; for example, Ateniese *et al.*'s scheme uses as many as 6*1024=6144 bits [1]. In Table 1, we compare the communication overheads of our ID-RDPC protocol, Ateniese *et al.*'s scheme and Sebé *et al.*'s scheme [4]. Our ID-RDPC scheme is the most efficient one in terms of communication.

5 Security Analysis

The security of the above ID-RDPC protocol is guaranteed by the following results.

Lemma 1: Suppose $\mathcal{A}$ is a $(T, \epsilon, Q_1, Q_2, Q_e, Q_T)$-forger against our $TagGen$ protocol, *i.e.*, $\mathcal{A}$ can forge a valid tag with probability ϵ within time T, where the groups $\mathcal{G}_1$ and $\mathcal{G}_2$ have prime order q. Then the CDH problem can be solved in $\mathcal{G}_1$ with probability ϵ' and within time T' satisfying $\epsilon' \geq \frac{1}{9}$ and

$$T' \leq \frac{23(Q_1 + Q_2)(T + T_{exp}(2Q_e + 4Q_T + 2Q_1 + Q_2))}{\mu\delta(1-\mu)^{Q_e+Q_T}(1-\delta)^{Q_2}\epsilon}$$

where T_{exp} denotes the time needed to perform a modular exponentiation in $\mathcal{G}_1$, μ and δ denote two real numbers from the interval $(0, 1)$, and Q_e, Q_1, Q_2, Q_T denote the number of queries to the Extract, H, h and Tag oracles, respectively.

Proof: Suppose that the set of stored index-message pairs is $\{(1, m_1), (2, m_2), \cdots, (n, m_n)\}$. We construct a probabilistic polynomial time Turing machine $\mathcal{F}$ which uses the attacker $\mathcal{A}$ as a sub-routine in order to solve a given instance of the CDH problem. Therefore, $\mathcal{F}$ will simulate the environment of $\mathcal{A}$.

Assuming the public data $(q, \mathcal{G}_1, \mathcal{G}_2, e, g, g^a, g^b)$ were given to $\mathcal{F}$, the goal of $\mathcal{F}$ is to compute the value g^{ab}. First, $\mathcal{F}$ defines $Y = g^a$ as the system public key and maintains three tables Tab$_1$, Tab$_2$ and Tab$_3$, which are initialized empty. Then $\mathcal{F}$ picks μ and δ from the interval $(0, 1)$, and answers all the queries that $\mathcal{A}$ makes.

Extract-Oracle. $\mathcal{A}$ makes an *Extract* query on the identity ID_i to $\mathcal{F}$. Then, $\mathcal{F}$ picks a bit $c_i \in \{0, 1\}$ satisfying the distribution $\Pr[c_i = 0] = \mu, \Pr[c_i = 1] = 1 - \mu$. According to the random value c_i, $\mathcal{F}$ proceeds as follows:

1) If $c_i = 0$, then $\mathcal{F}$ picks $\sigma_i, h_{1i} \in \mathcal{Z}_q$ independently and at random. $\mathcal{F}$ computes $R_i = g^{\sigma_i} \cdot Y^{-h_{1i}}$. Then, $\mathcal{F}$ stores the tuple (ID_i, R_i, h_{1i}) in the table Tab$_1$, and stores the tuple $(c_i, ID_i, R_i, h_{1i}, \sigma_i)$ in the table Tab$_3$.

2) If $c_i = 1$, then $\mathcal{F}$ picks $R_i, h_{1i} \in \mathcal{Z}_q$ independently and at random. Then, $\mathcal{F}$ stores the tuple (ID_i, R_i, h_{1i}) in the table Tab$_1$, and stores the tuple $(c_i, ID_i, R_i, h_{1i}, \sigma_i)$ in the table Tab$_3$, where $\sigma_i = \perp$ meaning that this value is unknown by $\mathcal{F}$.

H-Oracle. When $\mathcal{A}$ makes a query (ID_j, R_j) to H, $\mathcal{F}$ looks for (ID_j, R_j) in the table Tab$_1$. If it exists, then $\mathcal{F}$ answers the stored value h_{1j}. Otherwise, $\mathcal{F}$ picks at random $h_{1j} \in \mathcal{Z}_q$ and sends it to $\mathcal{A}$. Then, it stores the new tuple (ID_j, R_j, h_{1j}) in Tab$_1$.

h-Oracle. When $\mathcal{A}$ queries the random oracle h at an index i, algorithm $\mathcal{F}$ responds as follows:

1) If the query i already appears in the h-list (i, z_i, b_i, d_i) and $1 \leq i \leq n$ then algorithm $\mathcal{F}$ responds with $h(i) = z_i u^{-m_i}$;
2) Otherwise, $\mathcal{F}$ generates a random coin $d_i \in \{0, 1\}$ satisfying the distribution $\Pr[d_i = 0] = \frac{1}{q_s+1}$, $\Pr[d_i = 1] = 1 - \frac{1}{q_s+1}$.
3) Algorithm $\mathcal{F}$ picks a random $b_i \in \mathcal{Z}_q^*$. If $d_i = 1$, $\mathcal{F}$ computes $z_i = g^{b_i}$. If $d_i = 0$, $\mathcal{F}$ computes $z_i = (g^b)^{b_i}$.
4) Algorithm $\mathcal{F}$ adds the tuple (i, z_i, b_i, d_i) to the h-list and responds to $\mathcal{A}$ by setting $h(i) = z_i u^{-m_i}$.

TagGen-Oracle. Let (i, m_i) be a tag query issued by $\mathcal{A}$. Algorithm $\mathcal{F}$ responds to this query as follows:

1) Algorithm $\mathcal{F}$ runs the above algorithm for responding to h-list to obtain $h(i)$. Let (i, z_i, b_i, c_i) be the corresponding tuple in table Tab$_2$. If $d_i = 0$, then $\mathcal{F}$ reports failure and terminates.
2) When $d_i = 1$, define $\sigma_i = (g^a)^{b_i}$. Observe that $T_i = (g^a)^{b_i} = (g^{b_i})^a = (h(i)u^{m_i})^a$ under the public parameters (g^a, u). Then, $\mathcal{F}$ gives T_i to $\mathcal{A}$.

Output. Eventually algorithm $\mathcal{A}$ produces a index-message pair (j, m_j) such that no tag query was issued for (j, m_j). If there is no tuple on the h-list containing j, then $\mathcal{F}$ issues a query itself for $h(j)$ to ensure that such a tuple exists. We assume T_j is a valid tag on m_j under the given public key. Next, algorithm $\mathcal{F}$ finds the tuple (j, z_j, b_j, d_j) on the h-list. If $d_j = 1$ then $\mathcal{F}$ reports failure and terminates. Otherwise ($d_j = 0$), (T_j, m_j) satisfies the following checking equation

$$e(T_j, g) = e(h(j)u^{m_j}, R \cdot Y^{H(ID,R)})$$

Making use of the oracle-replay technique [27], $\mathcal{F}$ can obtain another tag $(\hat{T}_j, m_j)$ by using a different hash function $\hat{H}, \hat{h}$. Then, $\mathcal{F}$ can get

$$e(\hat{T}_j, g) = e(\hat{h}(j)u^{m_j}, R \cdot Y^{\hat{H}(ID,R)})$$

According to the simulation, $\mathcal{F}$ knows the corresponding $b_j, \hat{b}_j$ that satisfy $h(j) = (g^b)^{b_j}u^{-m_j}$, $\hat{h}(j) = (g^b)^{\hat{b}_j}u^{-m_j}$. Thus, $\mathcal{F}$ can get

$$e(T_j, g) = e((g^b)^{b_j}, R \cdot Y^{H(ID,R)})$$

$$e(\hat{T}_j, g) = e((g^b)^{\hat{b}_j}, R \cdot Y^{\hat{H}(ID,R)})$$

From the above, $\mathcal{F}$ can get

$$e(T_j \hat{T}_j^{\frac{-b_j}{\hat{b}_j}}, g) = e((g^b)^{b_j}, Y^{H(ID,R)-\hat{H}(ID,R)})$$

and finally obtain

$$g^{ab} = (T_j \hat{T}_j^{\frac{-b_j}{\hat{b}_j}})^{\frac{1}{b_j(H(ID,R)-\hat{H}(ID,R))}}$$

Probability analysis. In order to make $\mathcal{A}$ forge a valid tag, $\mathcal{F}$ must perfectly simulate the attack environment, *i.e.*, $\mathcal{F}$ cannot fail to answer $\mathcal{A}$'s queries. According to the simulation process, we know that the probability not to fail is $P_1 = \mu\delta(1-\mu)^{Q_e}(1-\delta)^{Q_2}$. Thus, the attacker $\mathcal{A}$ can forge a valid tag with probability

$$\hat{\epsilon} \geq \mu\delta(1-\mu)^{Q_e+Q_T}(1-\delta)^{Q_2}\epsilon$$

within the time $\hat{T} \leq T + T_{exp}(2Q_e + 4Q_T + 2Q_1 + Q_2)$. We use the oracle replay technique that was previously formalized [27]. $\mathcal{A}$ can get two different tags on the same message and randomness with the probability $\epsilon' \geq \frac{1}{9}$ within time $T' \leq 23(Q_1+Q_2)\hat{T}\hat{\epsilon}^{-1}$, *i.e.*, $T' \leq \frac{23(Q_1+Q_2)(T+T_{exp}(2Q_e+4Q_T+2Q_1+Q_2))}{\mu\delta(1-\mu)^{Q_e+Q_T}(1-\delta)^{Q_2}\epsilon}$. □

Since the CDH problem is known to be difficult, Lemma 1 above implies the following corollary.

Corollary 1 (Single-tag existential unforgeability): A single tag is existentially unforgeable if the CDH problem in $\mathcal{G}_1$ is hard.

Lemma 2: Based on the unforgeability of a single tag (Corollary 1), the grouped message-tag pair $(\hat{m}, T)$ can be forged only with negligible probability.

Proof: We will prove this theorem by contradiction. Suppose the forged message-tag pair $(\hat{m}, T)$ can pass the client's possession checking, *i.e.*,

$$\begin{aligned} & e(T, g) \\ = \;& e(\textstyle\prod_{j=1}^{c} h(\pi_{k_1}(j))u^{\hat{m}}, R\cdot Y^{H(ID,R)})e(\prod_{j=1}^{c} T_{i_j}^{a_j}, g) \\ = \;& e(\textstyle\prod_{j=1}^{c}[h(i_j)^{a_j}]u^{\sum_{j=1}^{c} a_j m_{i_j}}, R\cdot Y^{H(ID,R)}) \end{aligned}$$

where $a_j = f_{k_2}(j)$ are random numbers, $1 \leq j \leq c$. Then,

$$\textstyle\prod_{j=1}^{c} e(T_{i_j}, g)^{a_j} = \prod_{j=1}^{c} e(h(i_j)u^{m_{i_j}}, R\cdot Y^{H(ID,R)})^{a_j}$$

Suppose that the generator of $\mathcal{G}_2$ is d, and

$$e(T_{i_j}, g) = d^{x_j}, e(h(i_j)u^{m_{i_j}}, R\cdot Y^{H(ID,R)}) = d^{y_j}$$

then we can get

$$d^{\sum_{j=1}^{c} a_j x_j} = d^{\sum_{j=1}^{c} a_j y_j}, \qquad \sum_{j=1}^{c} a_j x_j = \sum_{j=1}^{c} a_j y_j, \qquad \sum_{j=1}^{c} a_j(x_j - y_j) = 0$$

According to Corollary 1, a single tag is existential unforgeable. So, there exist at least two different indices j such that $x_j \neq y_j$. Suppose there are $s \leq c$ pairs (x_j, y_j) such that $x_j \neq y_j$. Then, there exist q^{s-1} tuples $(a_1, a_2, \cdots, a_c)$ satisfying the above equation. As $(a_1, a_2, \cdots, a_c)$ is a random vector, the probability that the tuple satisfies the above equation is not greater than $q^{s-1}/q^c \leq q^{c-1}/q^c = q^{-1}$. This probability is negligible. □

Lemma 3: Assume some block-tag pair (m_l, T_l) is modified and PCS substitutes another valid block-tag pair $(m_{\hat{l}}, T_{\hat{l}})$ for (m_l, T_l). If PCS forges a response $(T, \hat{m})$ using the substituted block-tag, this response passes CheckProof only with negligible probability.

Proof: Without loss of generality, we assume that the client sends the challenge $chal = (c, k_1, k_2)$, where $l \in \{1, 2, \cdots, c\}$, $\hat{l} \notin \{1, 2, \cdots, c\}$.

Since the block-tag pair (m_l, T_l) is modified, the public cloud server may substitute another valid block-tag pair $(m_{\hat{l}}, T_{\hat{l}})$ for (m_l, T_l). Then,

$$T = \prod_{j=1, j\neq l}^{c} T_{i_j}^{a_j} T_{m_{i_{\hat{l}}}}^{a_l}, \qquad \hat{m} = \sum_{j=1, j\neq l}^{c} a_j m_{i_j} + a_l m_{i_{\hat{l}}}$$

where $a_i = f_{k_2}(i)$ is random number, $1 \le i \le c$.

If the forged response passed the checking, then

$$e(T, g) = e(\prod_{j=1}^{c} h(\pi_{k_1}(j)) u^{\hat{m}}, R \cdot Y^{H(ID,R)})$$

i.e.,

$$e(\prod_{j=1, j\neq l}^{c} T_{i_j}^{a_j} T_{m_{i_{\hat{l}}}}^{a_l}, g) = e(\prod_{j=1, j\neq l}^{c} [h(i_j)^{a_j}] u^{\sum_{j=1, j\neq l}^{c} a_j m_{i_j} + a_l m_{i_{\hat{l}}}}, R \cdot Y^{H(ID,R)})$$

Since the other block-tag pair is valid, we can get

$$e(T_{m_{i_{\hat{l}}}}^{a_l}, g) = e(h(i_l)^{a_l} u^{a_l m_{i_{\hat{l}}}}, R \cdot Y^{H(ID,R)})$$

According to the tag generation process, we know that

$$e(h(i_{\hat{l}})^{a_l} u^{a_l m_{i_{\hat{l}}}}, R \cdot Y^{H(ID,R)}) = e(h(i_l)^{a_l} u^{a_l m_{i_{\hat{l}}}}, R \cdot Y^{H(ID,R)})$$

i.e.,

$$h(i_{\hat{l}}) = h(i_l)$$

Since h is collision-free hash function, the probability of $h(i_{\hat{l}}) = h(i_l)$ is $\frac{1}{q}$, *i.e.*, it is negligible.

□

Corollary 1 states that an untrusted PCS cannot forge individual tags to cheat the client. Lemma 2 implies that the untrusted PCS cannot aggregate fake tags to cheat the client. Lemma 3 shows that the untrusted PCS cannot replace some tags to cheat the client. Hence, from the above results, we have the following claim.

Theorem 1: The proposed ID-RDPC protocol is unforgeable in the random oracle model if the CDH problem in $\mathcal{G}_1$ is hard.

Proof: Suppose that the set of stored index-message pairs is $\{(1, m_1)$, $(2, m_2)$, $\cdots$, $(n, m_n)\}$. Let $\mathcal{F}$ be a challenger and $\mathcal{A}$ be an adversary. If the public data $(q, \mathcal{G}_1, \mathcal{G}_2, e, g, g^a, g^b)$ was given to $\mathcal{F}$, the goal of $\mathcal{F}$ is to compute the value g^{ab}. First, $\mathcal{F}$ defines $Y = g^a$ as the system public key and maintains three tables Tab$_1$, Tab$_2$ and Tab$_3$, which are initialized empty. Then, $\mathcal{F}$ answers all the queries that $\mathcal{A}$ makes.

Extract-Oracle, H-Oracle, h-Oracle, TagGen-Oracle are the same as the corresponding procedures in Lemma 1.

Given the challenge $chal = (c, k_1, k_2)$, suppose the forged message-tag pair $(\hat{m}, T)$ can pass the client's possession checking, *i.e.*,

$$e(T, g) = e(\prod_{j=1}^{c} h(\pi_{k_1}(j))u^{\hat{m}}, R \cdot Y^{H(ID,R)}) \tag{1}$$

where $a_i = f_{k_2}(i)$ are random numbers, $1 \leq i \leq c$.

According to Corollary 1 and Lemma 1, we know that if some queried block-tag is corrupted, the verification formula (1) holds with negligible probability. Since the client does not store message blocks and the corresponding tags, $\mathcal{A}$ has the ability to substitute valid block-tag pairs for the corrupted block-tag pairs. According to Lemma 3, the substituted response can pass the verification only with negligible probability, *i.e.*, (1) holds with negligible probability.

Thus, in the random oracle, except with negligible probability no adversary can cause the verifier to accept the challenged proof if the challenged message-tag pair is corrupted or deleted.

□

Theorem 2: The proposed ID-RDPC scheme is $(\frac{t}{n}, 1 - (\frac{n-t}{n})^c)$-secure. Specifically, suppose that some PCS has stored n block-tag pairs $((m_1, T_1), (m_2, T_2), \cdots, (m_n, T_n))$ and has modified t block-

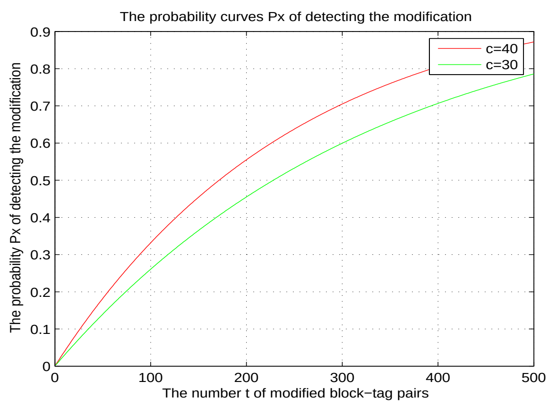

Fig. 3. Probability curve P_X of detecting the modification of several blocks

tag pairs. If the challenge is $chal = (c, k_1, k_2)$, the probability P_X of detecting the modification satisfies:

$$1 - (\frac{n-t}{n})^c \leq P_X \leq 1 - (\frac{n-c+1-t}{n-c+1})^c$$

Proof: The proof of this theorem is similar to the proof of deletion detection [1]. Assume that the PCS modifies t block-tag pairs out of the n block-tag pairs. Since $chal = (c, k_1, k_2)$, the number of different blocks for which the client asks proof in a challenge is c. Let X be a discrete random variable that is defined to be the number of blocks chosen by the client that match the blocks deleted by the PCS. The probability that at least one of the blocks picked by the client matches one of the blocks deleted by the PCS is denoted as P_X. We have:

$$\begin{aligned} P_X &= P\{X \geq 1\} = 1 - P\{X = 0\} \\ &= 1 - \frac{n-t}{n}\frac{n-1-t}{n-1} \cdot \ldots \cdot \frac{n-c+1-t}{n-c+1} \end{aligned}$$

Thus, we obtain

$$1 - (\frac{n-t}{n})^c \leq P_X \leq 1 - (\frac{n-c+1-t}{n-c+1})^c$$

This completes the proof of the theorem. □

Figure 3 depicts the probability P_X of detecting the modification of t blocks out of n as a function of t and for two different values of the number c of challenged blocks.

6 Conclusion

This paper formalizes an ID-RDPC model suitable for company-oriented cloud storage. We present the first ID-RDPC protocol proven secure under the assumption that the CDH problem is hard. In addition to the structural advantage of elimination of certificate management and verification, our ID-RDPC protocol also outperforms existing RDPC protocols in the PKI setting in terms of computation and communication.

Acknowledgments and disclaimer

This work was partly supported by the Spanish Government through projects TSI2007-65406-C03-01 “E-AEGIS”, TIN2011-27076-C03-01 “CO-PRIVACY” and CONSOLIDER INGENIO 2010 CSD2007-00004 “ARES”, by the Government of Catalonia under grant 2009 SGR 1135, by the European Commission under FP7 projects “DwB” and “Inter-Trust”, by Natural Science Foundation of Liaoning Province (No. 20102042), by China Post-doctor Science Fund (No. 20110490061) and by Program for Liaoning Excellent Talents in University (No. LJQ2011078), and by the NSF of China through projects 60970116, 61003214 and 61173154. The authors also acknowledge support by the Fundamental Research Funds for the Central Universities of China to Project 3103004, and Shaanxi Provincial Education Department through Scientific Research Program 2010JK727. The last author is partially supported as an ICREA-Acadèmia researcher by the Catalan Government. The authors are with the UNESCO Chair in Data Privacy, but this paper does not necessarily reflect the position of UNESCO nor does it commit that organization.

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song. Provable Data Possession at Untrusted Stores. *CCS'07*, pp. 598-609, 2007.

[2] G. Ateniese, R. DiPietro, L. V. Mancini, G. Tsudik. Scalable and Efficient Provable Data Possession. *SecureComm 2008*, article 9, 2008.

[3] C. C. Erway, A. Kupcu, C. Papamanthou, R. Tamassia. Dynamic Provable Data Possession. *CCS'09*, 213-222, 2009.

[4] F. Sebé, J. Domingo-Ferrer, A. Martínez-Ballesté, Y. Deswarte, J. Quisquater. Efficient Remote Data Integrity checking in Critical Information Infrastructures. *IEEE Transactions on Knowledge and Data Engineering*, 20(8):1034-1038, 2008.

[5] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, S. S. Yau. Efficient Provable Data Possession for Hybrid Clouds. *CCS'10*, 756-758, 2010.

[6] Y. Zhu, H. Hu, G.J. Ahn, M. Yu. Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage. *IEEE Transactions on Parallel and Distributed Systems*, 23(12):2231-224, 2012.

[7] R. Curtmola, O. Khan, R. Burns, G. Ateniese. MR-PDP: Multiple-Replica Provable Data Possession. *ICDCS'08*, 411-420, 2008.

[8] A. F. Barsoum, M. A. Hasan. Provable Possession and Replication of Data over Cloud Servers. CACR, University of Waterloo, Report2010/32,2010. Available at http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-32.pdf

[9] H. Wang. Proxy Provable Data Possession in Public Clouds. *IEEE Transactions on Services Computing*. To appear, available on-line at http://doi.ieeecomputersociety.org/10.1109/TSC.2012.35

[10] Z. Hao, N. Yu. A Multiple-Replica Remote Data Possession Checking Protocol with Public Verifiability. *2010 Second International Symposium on Data, Privacy, and E-Commerce*, 84-89, 2010.

[11] A. F. Barsoum, M. A. Hasan, On Verifying Dynamic Multiple Data Copies over Cloud Servers. *IACR eprint report 447, 2011*. Available at http://eprint.iacr.org/2011/447.pdf

[12] H. Wang, Y. Zhang. On the Knowledge Soundness of a Cooperative Provable Data Possession Scheme in Multicloud Storage. *IEEE Transactions on Parallel and Distributed Systems*. To appear, available on-line at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.16

[13] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li. Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing. *IEEE Transactions on Parallel And Distributed Systems* , 22(5):847-859, 2011.

[14] A. Juels, B. S. Kaliski Jr. PORs: Proofs of Retrievability for Large Files. *CCS'07*, 584-597, 2007.

[15] H. Shacham, B. Waters. Compact Proofs of Retrievability. *ASIACRYPT 2008*, LNCS 5350, 90-107, 2008.

[16] K. D. Bowers, A. Juels, A. Oprea. Proofs of Retrievability: Theory and Implementation. *CCSW'09*, 43-54, 2009.

[17] Q. Zheng, S. Xu. Fair and Dynamic Proofs of Retrievability. *CODASPY'11*, 237-248, 2011.

[18] Y. Dodis, S. Vadhan, D. Wichs1, Proofs of Retrievability via Hardness Amplification, *TCC 2009*, LNCS 5444, 109-127, 2009.

[19] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu. Zero-Knowledge Proofs of Retrievability. *Sci China Inf Sci*, 54(8):1608-1617, 2011.

[20] D. Boneh, M. Franklin. Identity-based Encryption from the Weil Pairing. *CRYPTO 2001*, LNCS 2139, 213-229, 2001.

[21] A. Miyaji, M. Nakabayashi, S. Takano. New Explicit Conditions of Elliptic Curve Traces for FR-reduction. *IEICE Transactions Fundamentals*, 5:1234-1243, 2001.

[22] D. Boneh, B. Lynn, H. Shacham. Short Signatures from the Weil Pairing. *ASIACRYPT 2001*, LNCS 2248, 514-532, 2001.

[23] S. Yu, K. Ren, W. Lou. FDAC: Toward Fine-grained Distributed Data Access Control in Wireless Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.*, 22(4):673-686, 2011.

[24] S. Yu, K. Ren, W. Lou. Attribute-based On-demand Multicast Group Setup with Membership Anonymity. *Computer Networks*, 54(3):377-386, 2010.

[25] P. S. L. M. Barreto, B. Lynn, M. Scott. Efficient Implementation of Pairing-based Cryptosystems. *Journal of Cryptology*, 17(4):321-334, 2004.

[26] H. W. Lim. On the Application of Identity-based Cryptography in Grid Security. *Ph.D. dissertation*, University of London, London, U.K., 2006.

[27] D. Pointcheval, J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361-396, 2000.

[28] C. Research, SEC 2: Recommended Elliptic Curve Domain Parameters, http://www.secg.org/collateral/sec_final.pdf