Identity-based Remote Data Integrity Checking with Perfect Data Privacy Preserving for Cloud Storage

Yong Yu, Man Ho Au*, Giuseppe Ateniese, Xinyi Huang, Yuanshun Dai, Willy Susilo and Geyong Min

Abstract— **Remote data integrity checking (RDIC) enables a data storage server, such as a cloud server, to prove to a verifier that it is actually storing a data owner's data honestly. To date, a number of RDIC protocols have been proposed in the literature, but almost all the constructions suffer from the issue of a complex key management, that is, they rely on the expensive public key infrastructure (PKI), which might hinder the deployment of RDIC in practice. In this paper, we propose a new construction of identity-based (ID-based) RDIC protocol by making use of key-homomorphic cryptographic primitive to reduce the system complexity and the cost for establishing and managing the public key authentication framework in PKI based RDIC schemes. We formalize ID-based RDIC and its security model including security against a malicious cloud server and zero knowledge privacy against a third party verifier. We then provide a concrete construction of ID-based RDIC scheme which leaks no information of the stored files to the verifier during the RDIC process. The new construction is proven secure against the malicious server in the generic group model and achieves zero knowledge privacy against a verifier. Extensive security analysis and implementation results demonstrate that the proposed new protocol is provably secure and practical in the real-world applications.**

Keywords: Cloud storage, data integrity, privacy preserving, identity-based cryptography.

I. Introduction

Cloud computing [1], which has received considerable attention from research communities in academia as well as industry, is a distributed computation model over a large pool of shared-virtualized computing resources, such as storage, processing power, applications and services. Cloud users are provisioned and de-provisioned recourses as they want in cloud computing environment. This kind of new computing represents a vision of providing computing services as public utilities like water and electricity. Cloud computing brings a number of advantages for cloud users. As examples, this include the following issues: (1) Users can avoid capital expenditure on hardware, software and services because they pay only for what they use; (2) Users can enjoy low management overhead and immediate access to a wide range of applications; and (3) Users can access their data wherever they are, rather than having to stay close to their computers.

However, there is a vast variety of barriers before cloud computing can be widely deployed. A recent survey by Oracle referred the data source from international data corporation enterprise panel, showing that security represents 87% of users' cloud fears[1]. One of the major security concerns of cloud users is the integrity of their outsourced files since they no longer physically possess their data and thus lose the control over their data. Moreover, the cloud server is not fully trusted and it is not mandatory for the cloud server to report data loss incidents. Indeed, to ascertain cloud computing reliability, the cloud security alliance (CSA) published an analysis of cloud vulnerability incidents. The investigation [2] revealed that the incident of data Loss & Leakage accounted for 25% of all incidents, ranked second only to "Insecure Interfaces & APIs". Take Amazon's cloud crash disaster as an example[2]. In 2011, Amazon's huge EC2 cloud services crash permanently destroyed some data of cloud users. The data loss was apparently small relative to the total data stored, but anyone who runs a website can immediately understand how terrifying a prospect any data loss is. Sometimes it is insufficient to detect data corruption when accessing the data because it might be too late to recover the corrupted data. As a result, it is necessary for cloud users to frequently check if their outsourced data are stored properly.

The size of the cloud data is huge, downloading the entire file to check the integrity might be prohibitive in terms of bandwidth cost, and hence, very impractical. Moreover, traditional cryptographic primitives for data integrity checking such as hash functions, authorisation code (MAC) cannot apply here directly due to being short of a copy of the original file in verification. In conclusion, remote data integrity checking for secure cloud storage is a highly desirable as well as a challenging research topic.

Blum proposed an auditing issue for the first time that enables data owners to verify the integrity of remote data without explicit knowledge of the entire data [3]. Recently, remote data integrity checking becomes more and more significant due to the development of distributed storage systems and online storage systems. Provable data possession (PDP) [4], [5] at untrusted stores, introduced by Ateniese et al., is a novel technique for "blockless validating" data integrity over remote

* Corresponding Author

Yong Yu, Yuanshun Dai and Geyong Min are with School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China. Email: yuyong@uestc.edu.cn

Yong Yu and Xinyi Huang are with Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350007, China.

Man Ho Au is with Department of Computing, The Hong Kong Polytechnic University, China.

Giuseppe Ateniese is with Department of Computer Science, Sapienza University of Roma, Italy

Willy Susilo is with the Center for Computer and Information Security Research, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia.

[1] IDC Enterprise Panel, 2010

[2] http://www.businessinsider.com/amazon-lost-data-2011-4

servers. In PDP, the data owner generates some metadata for a file, and then sends his data file together with the metadata to a remote server and deletes the file from its local storage. To generate a proof that the server stores the original file correctly, the server computes a response to a challenge from the verifier. The verifier can verify if the file keeps unchanged via checking the correctness of the response. PDP is a practical approach to checking the integrity of cloud data since it adopts a spot-checking technique. Specifically, a file is divided into blocks and a verifier only challenges a small set of randomly chosen clocks for integrity checking. According to the example given by Ateniese et al. [4], for a file with $10,000$ blocks, if the server has deleted 1% of the blocks, then a verifier can detect server's misbehavior with probability greater than 99% by asking proof of possession for only 460 randomly selected blocks. Ateniese et al. proposed two concrete PDP constructions by making use of RSA-based homomorphic linear authenticators. Due to its necessity and practicability, remote data integrity checking has attracted extensive research interest [7]–[10], [12]–[15], [17]–[20], [24], [25] in recent years. Shacham and Waters [7] proposed the notion of compact proofs of retrievability by making use of publicly verifiable homomorphic authenticators from BLS signature [36]. This scheme also relies on homomorphic properties to aggregate a proof into a small authenticator value and as a result, the public retrievability can be achieved.

Ateniese et al. [21] considered dynamic PDP scheme for the first time based on hash functions and symmetric key encryptions, which means the data owner can dynamically update their file after they store their data on the cloud server. The dynamics operation involves data insertion, modification, deletion and appending. This scheme [21] is efficient but has only limited number of queries and block insertion cannot explicitly be supported. Erway et al. [23] extended the PDP model to dynamic PDP model by utilizing rank-based authenticated skip lists. Wang et al. [15] improved the previous PDP models by manipulating the Merkle Hash Tree (MHT) for block tag authentication. A recent work due to Liu et al. [24] showed that MHT itself is not enough to verify the block indices, which may lead to replace attack. They gave top-down levelled multi-replica MHT based data auditing scheme for dynamic big data storage on the cloud.

In data integrity checking with public verifiability, an external auditor (or anyone) is able to verify the integrity of the cloud data. In this scenario, data privacy against the third party verifier is highly essential since the cloud users may store confidential or sensitive files say business contracts or medical records to the cloud. However, this issue has not been fully investigated. The "privacy" definition in the previous privacy-preserving public auditing scheme [13] requires that the verifier cannot recover the whole blocks from the responses generated by the cloud server. However, this definition is not strong enough, for example, it is vulnerable to dictionary attack. Wang et al. [25] proposed the notion of "zero-knowledge public auditing" to resist off-line guessing attack. However, a formal security model is not provided in this work. Yu et al. [26] recently enhanced the privacy of remote data integrity checking protocols for secure cloud storage, but their model works only in public key infrastructure (PKI) based scenario instead of the identity-based framework. Encrypting the file before outsourcing can partially address the data privacy issue but leads to losing the flexibility of the protocols, since privacy preserving RDIC protocols can be used as a building block for other primitives. For example, Ateniese et al. [27] proposed a framework for building leakage-resilient identification protocols in the bounded retrieval model from publicly verifiable proofs of storage that are computationally zero-knowledge, but in the identification schemes, even the encrypted files must stay private.

Currently, a majority of the existing RDIC constructions rely on PKI where a digital certificate is used to guarantee the genuine of a user's public key. These constructions incur complex key management procedures since certificate generation, certificate storage, certificate update and certificate revocation are time-consuming and expensive. There is a variety of standards, say the Internet X.509 PKI certificate policy and certification practices framework (RFC 2527), that cover aspects of PKI. However, it lacks predominant governing body to enforce these standards. Despite a certificate authority (CA) is often regarded as trusted, drawbacks in the security procedures of various CAs have jeopardized trust in the entire PKI on which the Internet depends on. For instance, after discovering more than 500 fake certificates, web browser vendors were forced to blacklist all certificates issued by DigiNotar, a Dutch CA, in 2011. An alternative approach to using a certificate to authenticate public key is identity-based cryptography [28], in which the public key of a user is simply his identity, say, his name, email or IP address. A trusted key distribution center (KGC) generates a secret key for each user corresponding to his identity. When all users have their secret keys issued by the same KGC, individual public keys become obsolete, thus removing the need for explicit certification and all associated costs. These features make the identity-based paradigm particularly appealing for use in conjunction with organization-oriented PDP. For example, a university purchases the cloud storage service for the staff and students, who have a valid E-mail address issued by the IT department of the university. All the members of the university can have a secret key provided by the KGC, say the IT department. The members of the university can store their data together with the meta-data of the file to the cloud. To ensure the data are stored properly, an auditor, a staff of IT department can check the integrity for any member with his E-mail address only, which can relieve the complex key management caused by PKI. The first ID-based PDP was proposed in [29] which converted the ID-based aggregate signature due to Gentry [30] to an ID-based PDP protocol. Wang [31] proposed another identity-based provable data possession in multi-cloud storage. However, their security model called unforgeability for identity-based PDP is not strong enough for capturing the property of soundness in the sense that, the challenged data blocks are not allowed for TagGen queries in this model, which indicates that the adversary cannot access the tags of those blocks. This is clearly not consistent with the real cloud storage where the cloud server is, in fact, storing the tags of all data blocks. Moreover, the concrete identity-based PDP protocol in [31] fails to achieve soundness, a basic security requirement of PDP schemes. The reason is that, the hashed value of each block is used for generating a tag of the block, as a consequence, a malicious cloud server can keep only the hash value of the blocks for generating a valid response to a challenge.

Our Contributions. The contributions of this paper are summarized as follows.

- In an ID-based signature scheme, anyone with access to the signer's identity can verify a signature of the signer. Similarly, in ID-based RDIC protocols, anyone knowing a cloud user's identity, say a third party auditor (TPA), is able to check the data integrity on behalf of the cloud user. Thus, public verifiability is more desirable than private verification in ID-based RDIC, especially for the resource constrained cloud users. In this case, the property of zero-knowledge privacy is highly essential for data confidentiality in ID-based RDIC protocols. Our first contribution is to formalize the security model of zero-knowledge privacy against the TPA in ID-based RDIC protocols for the first time.
- We fill the gap that there is no a secure and novel ID-based RDIC scheme to date. Specifically, we propose a concrete ID-based RDIC protocol, which is a novel construction that is different from the previous ones, by making use of the idea of a new primitive called asymmetric group key agreement [32], [33]. To be more specific, our challenge-response protocol is a two party key agreement between the TPA and the cloud server, the challenged blocks must be used when generating a shared key by the cloud server, which a response to a challenge from the TPA.
- We provide detailed security proofs of the new protocol, including the soundness and zero-knowledge privacy of the stored data. Our security proofs are carried out in the generic group model [34]. This is the first correct security proof of ID-based RDIC protocol. Thus, the new security proof method itself may be of independent interest.
- We show the practicality of the proposal by developing a prototype implementation of the protocol.

Organization: The rest of the paper are organized as follows. In Section II, we review some preliminaries used in ID-based RDIC construction. In Section III, we formalize the system model and security model of ID-based RDIC protocols. We describe our concrete construction of ID-based RDIC protocol in section IV. We formally prove the correctness, soundness and zero knowledge privacy of our ID-based RDIC protocol in section V. We report the performance and implementation results in section VI. Section VII concludes our paper.

II. Preliminaries

In this section, we review some preliminary knowledge used in this paper, including bilinear pairings and zero-knowledge proof.

A. Bilinear Pairing

A bilinear pairing [28] maps a pair of group elements to another group element. Specifically, let G_1, G_2 be two cyclic groups of order p. g_1 and g_2 denote generators of G_1 and G_2 respectively. A function $e : G_1 \times G_1 \to G_2$ is called a bilinear pairing if it has the following properties:

Bilinearity. For all $u, v \in G_1$ and $x, y \in Z_p$, $e(u^x, v^y) = e(u, v)^{xy}$ holds.

Non-Degeneracy. $e(g, g) \neq 1_{G_2}$ where 1_{G_2} is the identity element of G_2.

Efficient Computation. $e(u, v)$ can be computed efficiently (in polynomial time) for all $u, v \in G_1$.

B. Equality of Discrete Logarithm

Let G be a finite cyclic group such that $|G| = q$ for some prime q, and g_1, g_2 be generators of G. The following protocol [37] enables a prover *P* to prove to a verifier *V* that two elements Y_1, Y_2 have equal discrete logarithm to base g_1 and g_2 respectively.

Commitment. *P* randomly chooses $\rho \in Z_q$, computes $T_1 = g_1^\rho, T_2 = g_2^\rho$ and sends T_1, T_2 to *V*.

Challenge. *V* randomly chooses a challenge $c \in \{0, 1\}^\lambda$ and sends c back to *P*.

Response. *P* computes $z = \rho - cx \pmod q$ and returns z to *V*.

Verify. *V* accepts the proof if and only if $T_1 = g_1^z Y_1^c \wedge T_2 = g_2^z Y_2^c$ holds.

This protocol can be converted into a more efficient non-interactive version, which is denoted as $POK\{(x) : Y_1 = g_1^x \wedge Y_2 = g_2^x\}$, by replacing the challenge with the hash of the commitment, that is, $c = H(T_1||T_2)$, where H is a secure hash function.

C. ID-based Signature

An identity-based signature (IDS) scheme [38], [39] consists of four polynomial-time, probabilistic algorithms described below.

Setup(k). This algorithm takes as input the security parameter k and outputs the master secret key *msk* and the master public key *mpk*.

Extract(msk, ID). This algorithm takes as input a user's identity *ID*, the master secret key *msk* and generates a secret key *usk* for the user.

Sign(ID, usk, m). This algorithm takes as input a user's identity *ID*, a message *m* and the user's secret key *usk* and generates a signature σ of the message *m*.

Verify(ID, m, σ, mpk). This algorithm takes as input a signature σ, a message *m*, an identity *ID* and the master public key *mpk*, and outputs if the signature is valid or not.

III. System Model and Security Model

In this section, we describe the system model and security model of identity-based RDIC protocols.

A. ID-based RDIC System

Usually, data owners themselves can check the integrity of their cloud data by running a two-party RDIC protocol. However, the auditing result from either the data owner or the cloud server might be regarded as biased in a two-party scenario. The RDIC protocols with public verifiability enable anyone to audit the integrity of the outsourced data. To make the description of the publicly verifiable RDIC protocols clearly, we assume there exits a third party auditor (TPA) who has expertise and capabilities to do the verification work. With this in mind, the ID-based RDIC architecture is illustrated in Fig 1. Four different entities namely the KGC, the cloud user, the cloud server and the TPA are involved in the system. The KGC generates secret keys for all the users according to their identities. The cloud user has large amount of files to be stored on cloud without keeping a local copy, and the cloud server has significant storage space and computation resources and provides data storage services for cloud users. TPA has expertise and capabilities that cloud users do not have and is trusted to check the integrity of the cloud data on behalf of the cloud user upon request. Each entity has their own obligations and benefits respectively. The cloud server could be self-interested, and for his own benefits, such as to maintain a good reputation, the cloud server might even decide to hide data corruption incidents to cloud users. However, we assume that the cloud server has no incentives to reveal the hosted data to TPA because of regulations and financial incentives. The TPA's job is to perform the data integrity checking on behalf the cloud user, but the TPA is also curious in the sense that he is willing to learn some information of the users' data during the data integrity checking procedure.

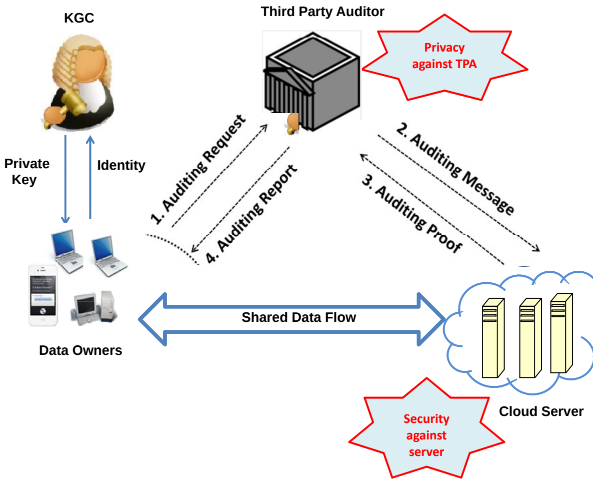


Fig. 1. The system model of identity-based RDIC

B. System Components and its Security

Six algorithms namely Setup, Extract, TagGen, Challenge, ProofGen and ProofCheck are involved in an identity-based RDIC system.

- Setup(1^k) is a probabilistic algorithm run by the KGC. It takes a security parameter k as input and outputs the system parameters $param$ and the master secret key msk.
- Extract$(param, msk, ID)$ is a probabilistic algorithm run by the KGC. It takes the system parameters $param$, the master secret key msk and a user's identity $ID \in \{0,1\}^*$ as input, outputs the secret key sk_{ID} that corresponds to the identity ID.
- TagGen$(param, F, sk_{ID})$ is a probabilistic algorithm run by the data owner with identity ID. It takes the system parameters $param$, the secret key of the user sk_{ID} and a file $F \in \{0,1\}^*$ to store as input, outputs the tags $\sigma = (\sigma_1, \cdots, \sigma_n)$ of each file block m_i, which will be stored on the cloud together with the file F.
- Challenge$(param, Fn, ID)$ is a randomized algorithm run by the TPA. It takes the system parameters $param$, the data owner's identity ID, and a unique file name Fn as input, outputs a challenge $chal$ for the file named Fn on behalf of the user ID.
- ProofGen$(param, ID, chal, F, \sigma)$ is a probabilistic algorithm run by the cloud server. It takes the system parameters $param$, the challenge $chal$, the data owner's identity ID, the tag σ, the file F and its name Fn as input, outputs a data possession proof P of the challenged blocks.
- ProofCheck$(param, ID, chal, P, Fn)$ is a deterministic algorithm run by the TPA. It takes the system parameters $param$, the challenge $chal$, the data owner's identity ID, the file name Fn and an alleged data possession proof P as input, outputs 1 or 0 to indicate if the file F keeps intact.

We consider three security properties namely completeness, security against a malicious server (soundness), and privacy against the TPA (perfect data privacy) in identity-based remote data integrity checking protocols. Following the security notions due to Shacham and Waters [7], an identity-based RDIC scheme is called secure against a server if there exists no polynomial-time algorithm that can cheat the TPA with non-negligible probability and there exists a polynomial-time extractor that can recover the file by running the challenges-response protocols multiple times. Completeness states that when interacting with a valid cloud server, the algorithm of ProofCheck will accept the proof. Soundness says that a cheating prover who can convince the TPA it is storing the data file is actually storing that file. We now formalize the security model of soundness for identity-based remote data integrity checking below, where an adversary who plays the role of the untrusted server and a challenger who represents a data owner are involved.

Security against the Server. This security game captures that an adversary cannot successfully generate a valid proof without possessing all the blocks of a user ID corresponding to a given challenge, unless it guesses all the challenged blocks. The game consists of the following phases [35].

- Setup: The challenger runs the Setup algorithm to obtain the system parameters $param$ and the master secret key msk, and forwards $param$ to the adversary, while keeps

msk confidential.

- Queries: The adversary makes a number of queries to the challenger, including extract queries and tag queries adaptively.
  1) Extract Queries: The adversary can query the private key of any identity ID_i. The challenger computes the private key sk_i by running the Extract algorithm and forwards it to the adversary.
  2) TagGen Queries: The adversary can request tags of any file F under the identity ID_i. The challenger runs the Extract algorithm to obtain the private key sk_i, and runs the TagGen algorithm to generate tags of the file F. Finally, the challenger returns the set of tags to the adversary.
- ProofGen: For a file F of which a TagGen query has been made, the adversary can undertake executions of the ProofGen algorithm by specifying an identity ID of the data owner and the file name Fn. The challenger plays the role of the TPA and the adversary $\mathcal{A}$ behaves as the prover during the proof generation. Finally, the adversary can get the output of P from the challenger when a protocol execution completes.
- Output: Finally, the adversary chooses a file name $Fn^\dagger$ and a user identity $ID^\dagger$. $ID^\dagger$ must not have appeared in key extraction queries and there exists a TagGen query with input $F^\dagger$ and $ID^\dagger$. The adversary outputs the description of a prover $P^\dagger$ which is ϵ-admissible defined below.

We say the cheating prover $P^\dagger$ is ϵ-admissible if it convincingly answers an ϵ fraction of integrity challenges. That is, $\Pr[(\mathcal{V}(param, ID^\dagger, Fn^\dagger) \rightleftharpoons \mathcal{P}^\dagger) = 1] \geq \epsilon$. The probability here is over the coins of the verifier and the prover. The adversary wins the game if it can successfully output a ϵ-admissible prover $P^\dagger$.

An ID-RDIC scheme is called ϵ-sound if there exists an extraction algorithm Extr such that, for every adversary $\mathcal{A}$, whenever $\mathcal{A}$, playing the soundness game, outputs an ϵ-admissible cheating prover $P^\dagger$ on identity $ID^\dagger$ and file name $Fn^\dagger$, Extr recovers $F^\dagger$ from $P^\dagger$, i.e., $\mathsf{Extr}(param, ID^\dagger, Fn^\dagger, P^\dagger) = F$, except possibly with negligible probability.

Perfect data privacy against the TPA. By "perfect data privacy" we mean that TPA achieves no information of the outsourced data, that is, whatever TPA learns, the TPA could have learned by himself without any interaction with the cloud server. We make use of a *simulator* to formalize this model as follows.

Definition 1. An identity-based remote data integrity checking protocol achieves perfect data privacy if for every cheating verifier TPA^*, there exists a polynomial time non-interactive simulator S for TPA^* such that for every valid public input $ID, chal, Tag$ and private input F, the following two random variables are computationally indistinguishable:

1) $view_{TPA^*}(Server_{R,chal,F,ID,Tag,TPA^*})$, where R denotes the random coins the protocol uses.
2) $S(chal, ID)$.

That is to say, the simulator S gets only the information of the public input and has no interactions with the server, but still manages to output a response indistinguishable from whatever TPA^* learns during the interaction.

IV. Our Construction

In this section, we provide a concrete construction of secure identity-based remote data integrity checking protocol supporting perfect data privacy protection. Our scheme works as follows. In the key extraction, we employ short signature algorithm due to Boneh et al. [36] to sign a user's identity $ID \in \{0,1\}^*$ and obtain the user's secret key. In TagGen, we invent a new algorithm to generate tags of file blocks which can be aggregated into a single element when computing a response to a challenge. In the challenge phase, the TPA challenges the cloud by choosing some indexes of the blocks and random values. In the proof generation, the cloud server computes a response using the challenged blocks, obtains the corresponding plaintext and forwards it to the TPA. The details of the proposed protocol are as follows.

Setup. On input a security parameter sp, the KGC chooses two cyclic multiplicative groups G_1 and G_2 with prime order q, where g is a generator of G_1. There exists a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. The KGC picks a random $\alpha \in Z_q^*$ as the master secret key and sets $P_{pub} = g^\alpha$. Finally, the KGC chooses three hash functions $H_1, H_2 : \{0,1\}^* \rightarrow G_1$ and $H_3 : G_2 \rightarrow \{0,1\}^l$. The system parameter is $(G_1, G_2, e, g, P_{pub}, H_1, H_2, H_3, l)$.

Extract. On input the master secret key α and a user's identity $ID \in \{0,1\}^*$, this algorithm outputs the private key of this user as $s = H_1(ID)^\alpha$.

TagGen. Given a file M named *fname*, the data owner firstly divides it into n blocks $m_1, \cdots, m_n$, where $m_i \in Z_q$ and then picks a random $\eta \in Z_q^*$ and computes $r = g^\eta$. For each block m_i, the data owner computes

$$\sigma_i = s^{m_i} H_2(fname||i)^\eta.$$

The tag for m_i is σ_i. The data owner stores the file M together with $(r, \{\sigma_i\}, \mathsf{IDS}(r||fname))$, to the cloud, where $\mathsf{IDS}(r||fname)$ is an identity-based signature [38], [39] from the data owner on the value $r||fname$.

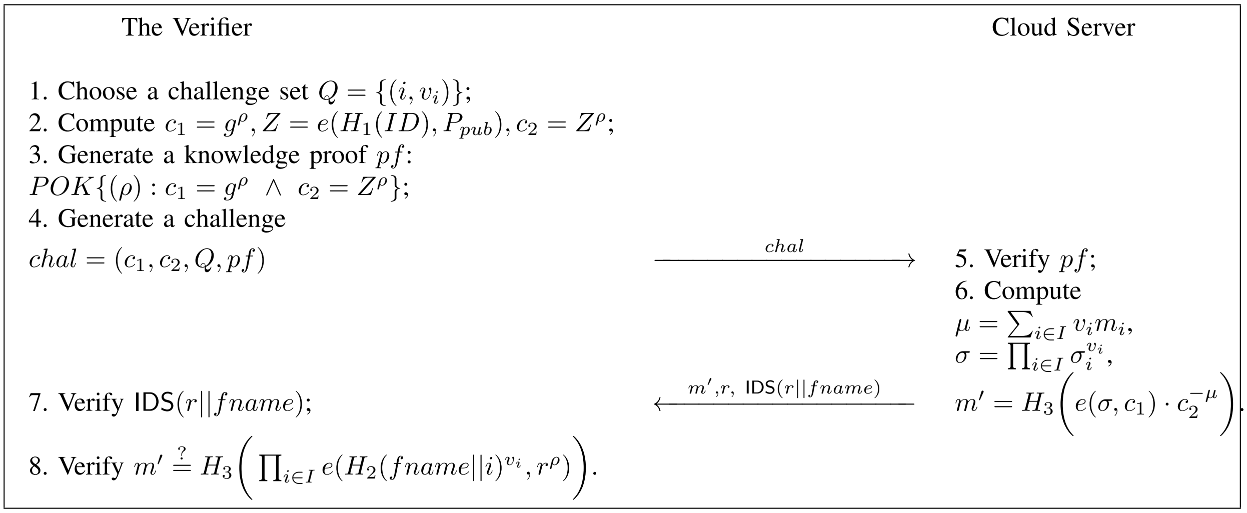
Challenge. To check the integrity of M, the verifier picks a random c-element subset I of the set $[1, n]$ and for each $i \in I$, chooses a random element $v_i \in Z_q^*$. Let Q be the set $\{(i, v_i)\}$. To generate a challenge, the verifier picks a random $\rho \in Z_q^*$, computes $Z = e(H_1(ID), P_{pub})$ and does the following.

1) Compute $c_1 = g^\rho, c_2 = Z^\rho$.
2) Generate a proof that :

$$pf = POK\{(\rho) : c_1 = g^\rho \ \wedge \ c_2 = Z^\rho\}.$$

The verifier sends the challenge $chal = (c_1, c_2, Q, pf)$ to the server.

GenProof. Upon receiving $chal = (c_1, c_2, Q, pf)$, the server first computes $Z = e(H_1(ID), P_{pub})$. Then, it verifies the proof pf. If it is invalid, the auditing aborts. Otherwise, the server computes $\mu = \sum_{i \in I} v_i m_i$, $\sigma = \prod_{i \in I} \sigma_i^{v_i}$ and $m' = H_3(e(\sigma, c_1) \cdot c_2^{-\mu})$ and returns $(m', r, \mathsf{IDS}(r||fname))$ as the response to the verifier.



The Verifier

Cloud Server

1. Choose a challenge set $Q = \{(i, v_i)\}$;
2. Compute $c_1 = g^{\rho}, Z = e(H_1(ID), P_{pub}), c_2 = Z^{\rho}$;
3. Generate a knowledge proof pf:
$POK\{(\rho) : c_1 = g^{\rho} \ \wedge \ c_2 = Z^{\rho}\}$;
4. Generate a challenge
$chal = (c_1, c_2, Q, pf)$

$\xrightarrow{chal}$

5. Verify pf;
6. Compute
$\mu = \sum_{i \in I} v_i m_i$,
$\sigma = \prod_{i \in I} \sigma_i^{v_i}$,
$m' = H_3\Big(e(\sigma, c_1) \cdot c_2^{-\mu}\Big)$.

$\xleftarrow{m', r,\ \mathsf{IDS}(r||fname)}$

7. Verify $\mathsf{IDS}(r||fname)$;
8. Verify $m' \overset{?}{=} H_3\Big(\prod_{i \in I} e(H_2(fname||i)^{v_i}, r^{\rho})\Big)$.

Fig. 2. Identity-based remote data integrity checking protocol

CheckProof. Upon receiving m' from the server, the verifier checks if $\mathsf{IDS}(r||fname)$ is a valid identity-based signature from the data owner on the message $r||fname$. If not, the proof is invalid. Otherwise, the verifier checks if

$$m' = H_3\Bigg(\prod_{i \in I} e(H_2(fname||i)^{v_i}, r^{\rho})\Bigg).$$

If the equality holds, the verifier accepts the proof; Otherwise, the proof is invalid.

V. Security Analysis of the New Protocol

In this section, we show that the proposed scheme achieves the properties of completeness, soundness and perfect data privacy preserving. Completeness guarantees the correctness of the protocol while soundness shows that the protocol is secure against an untrusted server. Perfect data privacy states that the protocol leaks no information of the stored files to the verifier.

A. Completeness

If both the data owner and the cloud server are honest, for each valid tag σ_i and a random challenge, the cloud server can always pass the verification. The completeness of the protocol can be elaborated as follows.

$$\begin{aligned}
m' &= H_3\Big(e(\sigma, c_1) \cdot c_2^{-\mu}\Big) \\
&= H_3\Bigg(\frac{e(\sigma, c_1)}{e(H(ID), P_{pub})^{\rho \sum_{i \in I} m_i v_i}}\Bigg) \\
&= H_3\Bigg(\frac{e(\prod_{i \in I} \sigma_i^{v_i}, c_1)}{e(s, g)^{\rho \sum_{i \in I} m_i v_i}}\Bigg) \\
&= H_3\Bigg(\frac{\prod_{i \in I} e(\sigma_i^{v_i}, c_1)}{\prod_{i \in I} e(s, c_1)^{m_i v_i}}\Bigg) \\
&= H_3\Bigg(\prod_{i \in I} e(\frac{\sigma_i}{s^{m_i}}, g^{\rho v_i})\Bigg) \\
&= H_3\Bigg(\prod_{i \in I} e(H_2(fname||i)^{\eta}, g^{\rho v_i})\Bigg) \\
&= H_3\Bigg(\prod_{i \in I} e(H_2(fname||i)^{v_i}, g^{\rho \eta})\Bigg) \\
&= H_3\Bigg(\prod_{i \in I} e(H_2(fname||i)^{v_i}, r^{\rho})\Bigg).
\end{aligned}$$

B. Soundness

In this part, we show that the probability of any *generic* algorithm in breaking the soundness of our protocol is negligible. A *generic* algorithm is an algorithm that does not exploit the algebraic structure of the underlying finite cyclic group. The generic group model is designed to capture the behavior of any *generic* algorithm. We briefly review the idea below.

Let p be a prime and $\xi_i : \mathbb{Z}_p \to \{0,1\}^{\lceil \log_2 p \rceil}$ for $i \in \{1, 2\}$ be two independent random encoding functions. The generic group $\mathbb{G}_i$ is represented as $\mathbb{G}_i = \{\xi_i(x) | x \in \mathbb{Z}_p\}$. Since a generic algorithm does not exploit the structure of the group, we say that given an element $\xi_i(x) \in \mathbb{G}_i$, nothing about the element except equality can be inferred. Two oracles, $\mathcal{O}_i$, $i \in \{1, 2\}$ are used to simulate the group action: for any element $\xi_i(a)$ and $\xi_i(b)$, the oracle query $\mathcal{O}_i$ on input $\xi_i(a)$,

$\xi_i(b)$ returns an element $\xi_i(a+b)$ (for multiplication) or $\xi_i(a-b)$ (for division). Another oracle, $\mathcal{O}_E$, is used to represent the pairing operation of $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Specifically, on input $\xi_1(a)$ and $\xi_1(b)$, oracle $\mathcal{O}_E$ returns $\xi_2(a * b)$.

Proof: After discussing the setting, we are going to show that our construction is secure against any *generic* algorithm in the random oracle model. Let $\mathcal{A}$ be a generic adversary making q_1, q_2, q_3 queries to the random oracles H_1, H_2, H_3 respectively, we show how to construct a simulator $\mathcal{S}$ which simulates the view of $\mathcal{A}$ in the generic group model and extracts from $\mathcal{A}$ the set of messages.

- *Settings.* In the generic group model and the random oracle model, the public parameters can be represented as:
$$\ell, q, \mathcal{O}_E, \mathcal{O}_1, \mathcal{O}_2, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \xi_1, \xi_\alpha,$$
where the first three oracles represent the pairing operation and group operations, and the next three oracles represent the hash operations, and ξ_1, ξ_α are the encodings of element g and P_{pub} respectively. They are chosen as random bit-strings by $\mathcal{S}$. $\mathcal{S}$ associates the encoding ξ_1 to a constant polynomial 1 and ξ_α to a multivariate polynomial α. Looking ahead, we shall see that at any stage, any group element will be associated with a multivariate polynomial. Specifically, any element presented to $\mathcal{A}$ will be associated with a multivariate polynomial in variables $(\alpha, \{I_i\}_{i=1}^{q_I}, \{f_i\}_{i=1}^{n*q_f}, \{\eta_i\}_{i=1}^{q_t}, \rho)$. We will discuss the rule of polynomial associations shortly.
- Hash Queries. For any query to $\mathcal{H}_1$ with input ID_k, $\mathcal{S}$ returns a random bit-string ξ_{I_k} and associates it to a multivariate polynomial I_k. For any query to $\mathcal{H}_2$ with input $fname_j||i$, $\mathcal{S}$ returns a random bit-string $\xi_{f_{j,i}}$ and associates it to a multivariate polynomial $f_{j,i}$. For any query to $\mathcal{H}_3$, $\mathcal{S}$ returns a random bit-string (which is not a group element).
- Group $\mathbb{G}_1$ operation (oracle $\mathcal{O}_1$). $\mathcal{A}$ presents to $\mathcal{S}$ two elements ξ_{F_1}, ξ_{F_2} and two integers, e_1, e_2. $\mathcal{S}$ looks up its history to locate the polynomial associated to elements ξ_{F_1} and ξ_{F_2} (assume they are F_1 and F_2 respectively). If such elements do not exist, it means ξ_{F_1} or ξ_{F_2} is not a valid group element and $\mathcal{S}$ rejects the operation. $\mathcal{S}$ computes $e_1F_1 + e_2F_2$ and the resulting polynomial is denoted as F_3. $\mathcal{S}$ lookup its history to check if there exists an element ξ in $\mathbb{G}_1$ which has been associated with F_3. If yes, it returns ξ. Otherwise, $\mathcal{S}$ picks a random bit-string ξ_{F_3} and associates it to polynomial F_3. It returns ξ_{F_3} to $\mathcal{A}$. Note that the element ξ_{F_3} represents the result of group operation ξ_{F_1} to the power of e_1 times ξ_{F_2} to the power of e_2.
- Group $\mathbb{G}_2$ operation (oracle $\mathcal{O}_2$). The handling is the same as that of $\mathcal{O}_1$ queries.
- Pairing operation (oracle $\mathcal{O}_E$). $\mathcal{A}$ presents to $\mathcal{S}$ two elements ξ_{F_1}, ξ_{F_2}. $\mathcal{S}$ lookup its history to locate the polynomial associated to elements ξ_{F_1} and ξ_{F_2} (assume they are F_1 and F_2 respectively). If such elements do not exist, it means ξ_{F_1} or ξ_{F_2} is not a valid group element and $\mathcal{S}$ rejects the operation. $\mathcal{S}$ computes $F_1 \cdot F_2$ and the resulting polynomial is denoted as F_3. $\mathcal{S}$ looks up its history to check if there exists an element ξ in $\mathbb{G}_2$ which has been associated with F_3. If yes, $\mathcal{S}$ returns ξ. Otherwise, $\mathcal{S}$ picks a random bit-string ξ_{F_3} and associates it to polynomial F_3. It returns ξ_{F_3} to $\mathcal{A}$. Note that the element ξ_{F_3} represent the result of the pairing operation on elements ξ_{F_1} and ξ_{F_3}.
- Extraction Query. $\mathcal{A}$ presents an identity ID to $\mathcal{S}$. $\mathcal{S}$ locates an $\mathcal{H}_1$ query with the same input. If not, it implicitly makes an $\mathcal{H}_1$ query. It obtains the element representing $H(ID)$, say ξ_{I_k} and its corresponding polynomial I_k. If there exists an element in $\mathbb{G}_1$, say ξ, which has been associated to the polynomial αI_k, $\mathcal{S}$ returns ξ. Otherwise, it picks a random bit-string $\xi_{\alpha I_k}$, associates it to the polynomial αI_k and returns it to $\mathcal{A}$.
- TagGen Query. $\mathcal{A}$ presents an identity ID and a file $M = (m_1, \ldots, m_n)$ to $\mathcal{S}$ with name $fname_j$. $\mathcal{S}$ locates an extract query with ID. If not, it implicitly makes an extraction query with ID as input. Assume the element $H(ID)$ is associated with polynomial I_k. It also obtains the set of elements $\{H_2(fname_j||i)\}$ ($\mathcal{S}$ makes those H_2 queries if they do not exist). Assume the element $H_2(fname_j||i)$ is associated with the polynomial $f_{j,i}$ $\mathcal{S}$ picks a random bit-string ξ_{η_j} and associates it with the polynomial η_j. For $i = 1$ to n, $\mathcal{S}$ computes the polynomial $F_{j,i} = \alpha I_k m_i + f_{j,i}\eta_j$. If an element $\xi_{j,i}$ has been associated with polynomial $F_{j,i}$, $\mathcal{S}$ returns $\xi_{j,i}$ as the tag $\sigma_{j,i}$ for block m_i for file $fname_j$. Otherwise, $\mathcal{S}$ picks a random bit-string $\xi_{j,i}$, associates it with $F_{j,i}$, and returns $\xi_{j,i}$ as the tag of the block m_i. The bit-string ξ_{η_j} is also given to $\mathcal{A}$ at this stage. Note that $(\xi_{\eta_j}, \xi_{j,1}, \ldots, \xi_{j,n})$ represents (r, T) for file M with name $fname_j$. An identity-based signature $\mathsf{IDS}(r||fname)$ on $r||fname_j$ is also given to $\mathcal{A}$.

At some stage $\mathcal{A}$ decides to be challenged on a certain target identity ID and a certain file $M = (m_1, \ldots, m_n)$ with name $fname$ under the restriction that $\mathcal{A}$ has never made an extraction query on ID. We can safely assume $\mathcal{A}$ has made a hash query on ID. We assume the element $H_1(ID)$ is associated with the polynomial I. $\mathcal{S}$ conducts a TagGen query with $\mathcal{A}$ using ID and M as input. Assume the elements returned to $\mathcal{A}$ are $(\xi_{\eta_j}, \xi_{j,1}, \ldots, \xi_{j,n})$. In the following we will drop the subscript j for clarity. Now, $\mathcal{S}$ plays the role of a challenger and $\mathcal{A}$ plays the role of the cloud server in the protocol.

- (Challenge). $\mathcal{S}$ picks a subset $\mathcal{I} \subset [1, n]$ and for each $i \in \mathcal{I}$, $\mathcal{S}$ picks v_i. $\mathcal{S}$ further picks a random bit-string ξ_{c_1} and associates to it polynomial ρ. It also picks another random bit-string ξ_Z and associates to it the polynomial αI. ξ_{c_1} and ξ_Z represents the group $\mathbb{G}_1$ element c_1 and the group $\mathbb{G}_2$ element Z respectively. It also chooses a random bit-string ξ_{c_2} and associates it to the polynomial $\alpha I \rho$. ξ_{c_1}, ξ_{c_2} are given to $\mathcal{A}$, along with a simulated proof pf.
- (Polynomial Evaluation). Recall that I is the polynomial associated with the element $H_1(ID)$. Let η be the polynomial associated with the element ξ_η. Further, let f_i be the polynomial associated with the element $H_2(fname||i)$.

Thus, the tag for m_i (the σ_i's) will be associated with polynomials $F_i \equiv m_i \alpha I + f_i \eta$. At this stage, $\mathcal{S}$ chooses at random the values for all variables except the following:

$$\alpha, I, \{f_i\}_{i \in \mathcal{I}}, \eta, \rho.$$

In other words, all polynomials given to $\mathcal{A}$ are multivariate polynomials in $c+4$ variables. The instantiation does not affect the view of the previous simulation as long as it does not create any "incorrect" relationship.

- (GenProof). Finally, $\mathcal{A}$ returns a value m', along with a group element r and its identity-based signature $\mathsf{IDS}(r||fname)$.

In the random oracle model, the only way for $\mathcal{A}$ to successfully return $m' = H_3(\prod_{i \in I} e(H_2(fname||i)^{v_i}, r^{\rho}))$ is to make a query to H_3 with an element ξ in group $\mathbb{G}_2$. Due to the unforgeability of the identity-based signature, we can safely assume r (represented by the element ξ_η used in the verification is the one given to $\mathcal{A}$ during the TagGen query.

For $\mathcal{A}$ to win with non-negligible probability, ξ must be associated with a polynomial F such that

$$F \equiv \sum_{i \in \mathcal{I}} v_i f_i \rho \eta.$$

It remains to show that in order to produce an element ξ whose associated polynomial satisfies the above equivalence relationship, $\mathcal{A}$ has to make a set of group operation queries which allows $\mathcal{S}$ to extract the set of constants $\{m_i\}$. Furthermore, these set of queries are made *after* the challenge phase. Firstly, note that for each query, $\mathcal{A}$ obtains at most a new group element and thus the total number of elements obtained by $\mathcal{A}$ is finite. Before the challenge phase, $\mathcal{A}$ has no element whose polynomial is associated with the variable ρ nor $I\alpha\rho$.

Before the challenge phase, each element in group $\mathbb{G}_1$ is associated with a polynomial of the following form:

$$A_0 + A_1\alpha + A_2\eta + A_3 I + \sum_{i \in \mathcal{I}} A_{4,i} f_i + \sum_{i \in \mathcal{I}} A_{5,i}(m_i I \alpha + f_i \eta)$$

where A's are coefficients known to $\mathcal{A}$. Likewise, each group element in group $\mathbb{G}_2$ is associated with a polynomial of the following form:

$$\sum_{j=1}^{k} c_j F_j * F'_j$$

where F_j and F'_j are polynomials associated with elements in group $\mathbb{G}_1$ and c_j being a constant. It is easy to see that polynomials obtained by $\mathcal{A}$ before the challenge will not satisfy the required relationship as the variable ρ is missing.

After the challenge phase, $\mathcal{A}$ has access to two additional elements whose polynomials are ρ in group $\mathbb{G}_1$ and $I\alpha\rho$ in group $\mathbb{G}_2$ respectively. From this, all group $\mathbb{G}_2$ elements obtained by $\mathcal{A}$ are associated with a polynomial of the following form:

$$\begin{aligned}
& A_0 + A_1\rho + A_2\alpha + A_3\eta + A_4 I + \sum_{i \in \mathcal{I}} A_{5,i} f_i + \\
& \sum_{i \in \mathcal{I}} A_{6,i}(m_i I \alpha + f_i \eta) + A_7 I \alpha \rho + \\
& B_1 \rho^2 + B_2 \rho\alpha + B_3 \rho\eta + B_4 \rho I + \\
& \sum_{i \in \mathcal{I}} B_{5,i} f_i \rho + \sum_{i \in \mathcal{I}} B_{6,i}(m_i I \alpha \rho + f_i \eta \rho) + B_7 I \alpha \rho^2 + \\
& C_1 \alpha^2 + C_2 \alpha\eta + C_3 \alpha I + \sum_{i \in \mathcal{I}} C_{4,i} f_i \alpha + \\
& \sum_{i \in \mathcal{I}} C_{5,i}(m_i I \alpha^2 + f_i \eta \alpha) + C_6 I \alpha^2 \rho + \\
& D_1 \eta^2 + D_2 I \eta + \sum_{i \in \mathcal{I}} D_{3,i} f_i \eta + \sum_{i \in \mathcal{I}} D_{4,i}(m_i I \alpha \eta + f_i \eta^2) + \\
& D_5 I \alpha \rho \eta + \\
& E_1 I^2 + \sum_{i \in \mathcal{I}} E_{2,i} f_i I + \sum_{i \in \mathcal{I}} E_{3,i}(m_i I^2 \alpha + f_i I \eta) + E_4 I^2 \alpha \rho + \\
& \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} F_{1,i,j} f_i f_j + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} F_{2,i,j}(m_i I \alpha f_j + f_i f_j \eta) + \\
& \sum_{i \in \mathcal{I}} F_{3,i} I \alpha \rho f_i + \\
& \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} G_{1,i,j}(m_1 I \alpha + f_i \eta)(m_j I \alpha + f_j \eta) + \\
& \sum_{i \in \mathcal{I}} G_{2,i}(m_i I^2 \alpha^2 \rho + f_i \eta I \alpha \rho) + \\
& H_1 I^2 \alpha^2 \rho^2
\end{aligned}$$

where all coefficients are known to $\mathcal{A}$ and $\mathcal{S}$. Equating the coefficients with F, all coefficients are 0 except $B_{6,i}$ and A_7. Specifically, $B_{6,i} = v_i$ for all $i \in \mathcal{I}$ and $A_7 = \sum_{i \in \mathcal{I}} m_i v_i$.

In other words, $\mathcal{S}$ can extract from $\mathcal{A}$ the terms A_7 which is $\sum_{i \in \mathcal{I}} m_i v_i$. With $|\mathcal{I}|$ interactions, $\mathcal{S}$ can compute all the values of m_i from $\mathcal{A}$. Furthermore, observe that the operations that leads to this coefficient A_7 has to be issued after the challenge since they are related to the polynomial $I\alpha\rho$. This completes the proof.

■

C. Perfect Data Privacy Preserving

To prove that the scheme preserve data privacy, we show how to construct a simulator $\mathcal{S}$, having blackbox-access to verifier V, can simulate the remote data integrity checking protocol without the knowledge of the data file blocks $\{m_i\}$ nor their corresponding $\{\sigma_i\}$[3].

We assume $IDS(r||fname), r, fname$ is given to $\mathcal{S}$. In other words, our protocol does not preserve the privacy of the file name nor the parameter r. Since $r = e(g, g)^{\eta}$, where η is a random value chosen by the data owner, it is reasonable to say r does not contain the information of the file blocks. Below we show how $\mathcal{S}$ answers the challenge given by a verifier V.

Upon receiving the challenge $chal$ from V, $\mathcal{S}$ parses $chal$ as (c_1, c_2, Q, pf). Next, $\mathcal{S}$ extracts from V the value ρ. Due

[3] Since $\{\sigma_i\}$ also contains information about the file block m_i.

to the soundness of pf, $\mathcal{S}$ obtains ρ such that $c_1 = g^{\rho}$ and $c_2 = e(H_1(ID), P_{pub})^{\rho}$. $\mathcal{S}$ parses Q as $\{(i, v_i)\}$ and computes $m' = H_3(\prod_{i \in \mathcal{I}} e(H_2(fname||i)^{v_i}), r^{\rho})$. $\mathcal{S}$ outputs $(m', r, IDS(r||fanme))$ as the response to this challenge.

For each challenge (c_1, c_2, Q, pf), there is a unique value m' that is valid. Thus, the above simulation is perfect.

VI. Performance and Implementation

In this section, we first numerically analyze the cost of the new protocol, and then report its experimental results.

A. Numerical Analysis

We provide a numerical analysis of costs regarding computation, communication and storage of the proposed protocol in this part.

Computation cost. We present the computation cost from the viewpoint of the KGC, the data owner, the cloud server and the verifier (TPA). For simplicity, we denote by E_{G1}, E_{G2} the exponentiations in G_1 and G_2, M_{G1}, M_{G2} the multiplication in G_1 and G_2, P the pairing computation and H the map-to-point hash function respectively. We ignore the ordinary hash functions in numerical analysis since the cost of these functions is negligible compared with other operations.

The primary computation of the KGC is generating system parameters and private key for each user. Thus, the main computational cost of KGC is $2E_{G1} + 1H$. The dominated computation of data owner is generating tags for file blocks as $\sigma_i = s^{m_i} H_2(fname||i)^{\eta}$, which is the most expensive operation in the protocol but fortunately it can be done offline. For a file with n blocks, the cost of the data owner is $(2n+1)E_{G1} + nH$. The main cost of the verifier is generating a challenge and checking the validity of a proof. The verifier needs to perform 1 pairing operation, and 6 exponentiations in G_1 to generate a challenge when using the proof of equality of discrete logarithm given in [37]. When checking a proof, the cost of the verifier is $(c+1)E_{G1} + cP + cH + (c-1)E_{G2}$. The main computation cost of generating a proof by the cloud server is calculating the aggregation of σ_i, that is $\sigma = \prod_{i \in I} \sigma_i^{v_i}$, and the total cost is $2P + (2c-1)M_{G1} + E_{G2} + M_{G2}$.

Communication cost. In the challenge phase, the verifier sends (c_1, c_2, Q, pf) to the server, where $Q = \{(i, v_i)\}$ denotes the challenge set. In practice, we can employ a pseudo-random function θ and a pseudo-random permutation ψ as the public parameters. The verifier only needs to include a key k_1 of θ and a key k_2 of ψ in the challenge, instead of the challenge set Q, which can reduce the communication cost significantly. In this case, the communication cost is of binary length $log_2 c_1 + log_2 c_2 + log_2 k_1 + log_2 k_2 + log_2 pf$. In the response phase, the server returns $(m', r, IDS(r \parallel fname))$ as the response to the verifier. An identity-based signature usually contains two points (320 bits) of elliptic curves, thus the communication cost of the response is of binary length $l + log_2 r + 320$.

Storage cost. Regarding the storage cost of the cloud server, the data owner and the verifier, since identity-based data auditing schemes are publicly verifiable, both the data and the tags are stored on the cloud server. An identity-based digital signature algorithm is used to protect the commitment r from being tampered by external and internal adversaries. In this case, what stored on the cloud are as follows.

Data	Tags	r||IDS(r,fname)

The data owner needs to store nothing in the context of public verifiability. With the knowledge of the identity information of the data owner, a verifier is able to check the integrity of the data on behalf the data owner. Thus, what the verifier needs to store is the challenge set Q and the value ρ randomly selected by himself. The storage cost of a verifier can be reduced to $log_2 k_1 + log_2 k_2 + log_2 q$ bits. The storage cost of the block tags is upper bounded by $\lceil log_2(M)/log_2 q \rceil 160$ bits when employing proper elliptic curves [28]. The cloud server hosts the data, the tags, r and its signature. As a consequence, the storage cost of the cloud server is upper bounded by $log_2 M + \lceil log_2(M)/log_2 q \rceil 160 + log_2 r + len(IDS(r \parallel fname))$.

B. Implementation Results

The implementation was conducted with pbc-0.5.13 [40] with pbc wrapper-0.8.0 [41] on Intel i7-4700MQ CPU @ 2.40GHz. The memory is always sufficient since the scheme only requires a polynomial space. In our implementation, we made use of parameter a.param, one of the standard parameter settings of pbc library. Parameter a.param provides a symmetric pairing with the fastest speed among all default parameters. The implementation time overheads of the protocol are displayed in the following two parts.

In the first part, we setup a file of a constant size of 1 MB, and observe the impact of the number of challenged blocks in terms of the time cost. In our setting, the size of a data block is bounded by the group order p, i.e., 160 bits. Hence, we have $50,000$ blocks in total. This implies that the timing results for Setup, Extract and TagGen steps are constant for this part. See Table I for more details. We can see that both the Setup algorithm and the Extract algorithm are extremely fast. The Setup algorithm picks some random values and compute a modular exponentiation in G_1, which costs 4.8ms, and the Extract algorithm needs to perform one modular exponentiation in G_1 for generating the private key of a cloud user, which cost 0.1ms. The TagGen algorithm is expensive and we show that the TagGen timing result consists of two phases, an off-line phase, where the data owner can preprocess $H_2(fname||i)^{\eta}$ without knowing the actual data; and an on-line phase, where the data owner needs to compute s^{m_i} for each data block. (Note that since all those exponentiations are carried out with a same base, we can use a lookup table to accelerate those single-base multiple-exponent operations.) The time cost of off-line computation of generating tags for 1 MB file is 241.9 seconds while the on-online time cost is 20.3 seconds. This is an acceptable result compared with the previous preprocessing result in [5].

We then increase the number of challenged blocks from 50 to 1000 with an increment of 50 for each test to see the time cost of Challenge, GenProof and CheckProof steps. As one shall see from Figure 3, the timing cost of those three

Setup	Extract	TagGen: off-line	TagGen: on-line	Challenge	GenProof	CheckProof
4.8 ms	0.1 ms	241.9 second	20.3 second	351 ns per challenge	1.3 ms per challenge	6.6 ms per challenge

TABLE I
SUMMERISE OF THE TIME COST FOR A 1 MB FILE

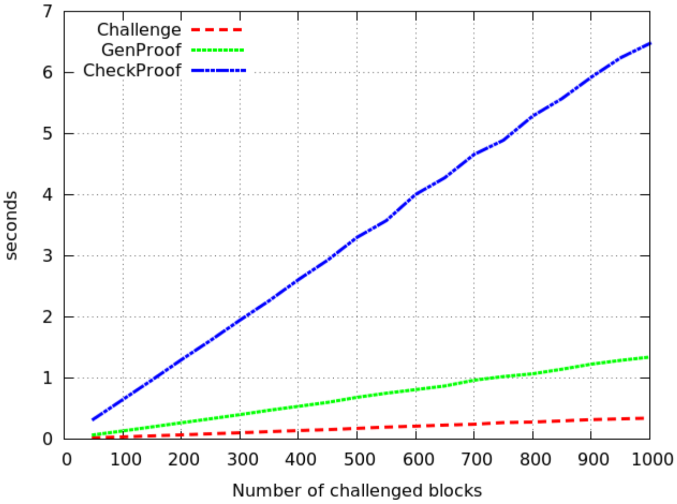


Fig. 3. Increasing number of challenges for fixed size of file

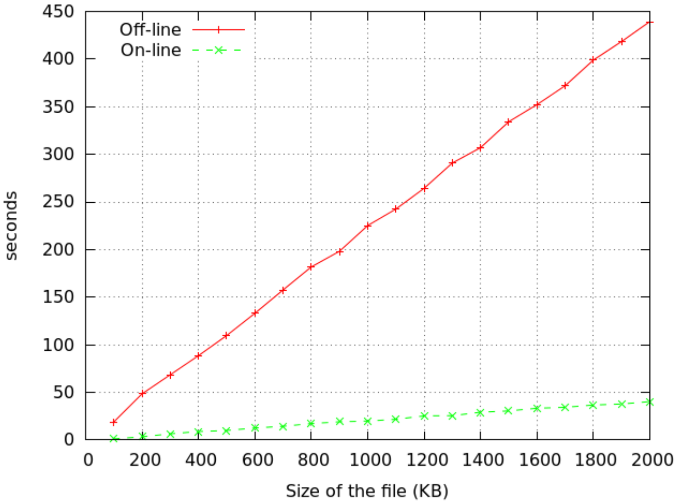


Fig. 4. Tag generation time for increased size of files

parts increases with the increase of the number of challenges. This is consistent with the empirical analysis because when the number of challenged block rises, more random values $\{i, v_i\}$ need to be selected in the challenge algorithm, and the cloud server has an increasing computations of $\sigma_i^{v_i}$ while the TPA has more exponentiations, multiplication operations and pairing computations in $\prod_{i \in I} e(H_2(fname||i)^{v_i}, r^{\rho})$. But it is fair to say that the proposed identity-based RDIC protocol is efficient for both the cloud server and the verifier. For example, let us investigate a point on the curve. Ateniese et al. [4] demonstrated that, if the cloud server has polluted 1% of the blocks, then the verifier can challenge 460 blocks in order to achieve the probability of server misbehavior detection of at least 99%. We can see that it costs the verifier only about 3.0 seconds to verify a response and the server 0.7 seconds to generate a response when challenging 460 blocks.

In the second part, we test the most expensive algorithm TagGen of the protocol by increasing the file size from 200 KB to 2 MB, that is, from $10,000$ blocks to $100,000$ blocks accordingly, and record the time for TagGen. As expected, both on-line and off-line time to generate tags for a given file increases almost linearly with the increase of the file size. Figure 4 gives more details. The implementation shows that generating tags is more expensive than other parts but fortunately, computing tags for a file is a one time task, as compared to challenging the outsourced data, which will be done repeatedly. Since the cloud users can do the off-line work completely parallelizable in advance, we pay only attention to the on-line cost. We can see that the efficiency of TagGen of our protocol is comparable to that of the existing well-known schemes, say [5]. To generate tags for a 2 MB file, it costs almost 42 seconds. As such, one shall be able to anticipate the time cost of generating tags for any size of files.

VII. CONCLUSION

In this paper, we investigated a new primitive called identity-based remote data integrity checking for secure cloud storage. We formalized the security model of two important properties of this primitive, namely, soundness and perfect data privacy. We provided a new construction of of this primitive and showed that it achieves soundness and perfect data privacy. Both the numerical analysis and the implementation demonstrated that the proposed protocol is efficient and practical.

ACKNOWLEDGEMENTS. This work is supported by the National Natural Science Foundation of China (61501333,61300213,61272436,61472083), Fok Ying Tung Education Foundation (141065), Program for New Century Excellent Talents in Fujian University (JA14067).

REFERENCES

[1] P. Mell, T. Grance, Draft NIST working definition of cloud computing, Reference on June. 3rd, 2009. http://csrc.nist.gov/groups/SNC/cloud-computing/index.html.

[2] Cloud Security Alliance. Top threats to cloud computing. http://www.cloudsecurityalliance.org, 2010.

[3] M. Blum, W. Evans, P.Gemmell, S. Kannan, M. Naor, Checking the correctness of memories. Proc. of the 32nd Anual Symposium on Foundations fo Vomputers, SFCS 1991, pp. 90–99, 1991.

[4] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, D. X. Song, Provable data possession at untrusted stores. ACM Conference on Computer and Communications Security, 598-609, 2007.

[5] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. N. J. Peterson, and D. Song, Remote data checking using provable data possession. *ACM Trans. Inf. Syst. Secur.*, 14, 1–34, 2011.

[6] A.Juels, and B. S. K. Jr. Pors, proofs of retrievability for large files.*Proc. of CCS 2007*, 584-597, 2007.

[7] H. Shacham, and B. Waters, Compact proofs of retrievability. *Proc. of Cryptology-ASIACRYPT 2008*, LNCS 5350, pp. 90-107, 2008.

[8] G. Ateniese, S. Kamara, J. Katz, Proofs of storage from homomorphic identification protocols. Proc. of ASIACRYPT 2009, 319-333, 2009.

[9] A. F. Barsoum, M. A. Hasan, Provable multicopy dynamic data possession in cloud computing systems, *IEEE Trans. on Information Forensics and Security*, 10(3): 485–497, 2015.

[10] J. Yu, K. Ren, C. Wang, V. Varadharajan, Enabling cloud storage auditing with key-exposure resistance, *IEEE Trans. on Information Forensics and Security*, 10(6): 1167–1179, 2015.

[11] J. Liu, K. Huang, H. Rong, H. M. Wang, Privacy-preserving public auditing for regenerating-code-based cloud storage, *IEEE Trans. on Information Forensics and Security*, 10(7): 1513–1528, 2015.

[12] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, Enabling public verifiability and data dynamics for storage security in cloud computing. *Proc. of ESORICS2009*, LNCS 5789, 355–370, 2009.

[13] C. Wang, Q. Wang, K. Ren, and W. Lou, Privacy-preserving public auditing for data storage security in cloud computing. *Proc of IEEE INFOCOM 2010*, 525–533, 2010.

[14] C. Wang, K. Ren, W. Lou, and J. Li, Toward publicly auditable secure cloud data storage services. *IEEE Network*, 24, 19-24, 2010.

[15] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, Enabling public audibility and data dynamics for storage security in cloud computing. *IEEE Trans. Parallel Distrib. Syst.*, 22, 847-859, 2011.

[16] C. Wang, S. S.Chow, Q. Wang, K. Ren, and W. Lou, Privacy-preserving public auditing for secure cloud storage. *IEEE Trans. on Computers*, 62, 362-375, 2013.

[17] K. Yang, and X. Jia. An efficient and secure dynamic auditing protocol for data storage in cloud computing. *IEEE Trans. on Parallel and Distributed Systems*, 24(9): 1717-1726, 2013.

[18] Y. Zhu, G. J. Ahn, H. Hu, S. S. Yau, H. G. An, C. J. Hu, Dynamic audit services for outsourced storages in Clouds. *IEEE Trans. Services Computing*, 6(2): 227-238, 2013.

[19] Y. Zhu, H. Hu, G. J. Ahn, S. S. Yau, Efficient audit service outsourcing for data integrity in clouds. *Journal of Systems and Software*, 85(5): 1083-1095, 2012.

[20] H. Wang, Y. Zhang, On the knowledge soundness of a cooperative provable data possession scheme in multicloud storage, *IEEE Trans. on Parallel and Distributed System*, 25(1): 264–267, 2014.

[21] G. Ateniese, R. D. Pietro, L. V. Mancini, G. Tsudik, Scalable and efficient provable data possession, Proc. of SecureComm 2008, Article No. 9 ,doi:10.1145/1460877.1460889.

[22] C. Wang, Q. Wang, K. Ren, W. Lou. Ensuring data storage secruty in cloud computing, Proc. of IWQoS 2009, 1-9, 2009.

[23] C. Erway, A. Kupcu, C. Papamanthou, R. Tamassia, Dynamic provable data possession, Proc. of CCS 2009, 213-222, 2009.

[24] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, J. Chen, MuR-DPA: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud. *IEEE Trans. on Computers*, 64(9): 2609–2622, 2015.

[25] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, Privacy-preserving public auditing for secure cloud storage, *IEEE Trans. on Computers*, 62, 362–375, 2013.

[26] Y. Yu, M H Au, Y. Mu, S. Tang, J. Ren, W. Susilo, and L. Dong, Enhanced privacy of a remote data integrity checking protocol for secure cloud storage, *International Journal of Information Security*, 14(4): 307–318, 2015.

[27] G. Ateniese, A. Faonio, S. Kamara, Leakage-resilient identification schemes from zero-knowledge proofs of storage. IMA Int. Conf. 2015, 311-328, 2015.

[28] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, Proc. of CRYPTO 2001, LNCS 2139, 213–229, 2001.

[29] J. N. Zhao, C. X. Xu, F. G. Li, and W. Z. Zhang, Identity-Based Public Verification with Privacy-Preserving for Data Storage Security in Cloud Computing, *IEICE Transactions*, 96-A(12), 2709-2716, 2013.

[30] G. Gentry, Z. Ramzan, Identity-Based aggregate signature, *Proc. of Public Key Cryptography 2006*, LNCS 3958, 257-271, 2006.

[31] H. Wang, Identity-based distributed provable data possession in multicloud storage, *IEEE Trans. on Service Computing*, 8(2), 328–340, 2015.

[32] Q. Wu, Y. Mu, W. Susilo, B. Qin, J. Domingo-Ferrer, Asymmetric group key agreement. Proc. of Eurocrypt 2009, LNCS 5479, 153–170, 2009.

[33] L. Zhang, Q. Wu, J. Domingo-Ferrer, B. Qin, Z. Dong, Round-efficient and sender-unrestricted dynamic group key agreement protocol for secure group communications. IEEE Transactions on Information Forensics and Security, 10(11): 2352–2364, 2015.

[34] V. Shoup, Lower bounds for discrete logarithms and related problems, Proc. of Eurocrypt 1997, 256–266, 1997.

[35] Y. Yu, Y. Zhang, Y. Mu, W. Susilo, Provably Secure Identity based Provable Data Possession. Proc. of ProvSec 2015, LNCS 9451, 1–16, 2015.

[36] D. Boneh, B.Lynn, and H. Shacham, Short signatures from the weil pairing. *Journal of Cryptology*, 17, 297–319, 2004.

[37] D. Chaum, T. P. Pedersen, Wallet databases with observers. Proc. of Crypto 1992, 89–105, 1993.

[38] J. C. Cha, and J. H. Cheon, An identity-based signature from gap Diffie-Hellman groups, in: *Processing of PKC 3003*, LNCS 2567, 2003, pp. 18-30.

[39] F. Hess, Efficient identity based signature schemes based on pairings, in: *Processing of SAC 2002*, LNCS 2595, 2003, pp. 310-324.

[40] B. Lynn, The Pairing-Based Cryptography Library (0.5.13). online: `http://crypto.stanford.edu/pbc/`.

[41] A. Kate, The Pairing-Based Cryptography (PBC) Library - C++ Wrapper Classes (0.8.0). online: `http://crysp.uwaterloo.ca/software/PBCWrapper/`.