A RESEARCH IN IMPLEMENTING THE REMOTE DATA INTEGRITY CHECKING IN PUBLIC CLOUD FOR IDENTITY-BASED PROXY-ORIENTED DATA UPLOADING

YADDANAPUDI NAGA SANDHYA RANI
Krishna Chaitanya Institute of Technology & Sciences,
Markapur, Prakasam(dist),A.P.

M. GNANA VARDHAN
Associate Professor, Krishna Chaitanya Institute of Technology & Sciences,
Markapur, Prakasam(dist),A.P.
Vardhan.nec@gmail.com

ABSTRACT:

More and more clients would like to store their data to public cloud servers (PCSs) along with the rapid development of cloud computing. New security problems have to be solved in order to help more clients process their data in public cloud. When the client is restricted to access PCS, he will delegate its proxy to process his data and upload them. On the other hand, remote data integrity checking is also an important security problem in public cloud storage. It makes the clients check whether their outsourced data are kept intact without downloading the whole data. From the security problems, we propose a novel proxy-oriented data uploading and remote data integrity checking model in identity-based public key cryptography: identity-based proxy-oriented data uploading and remote data integrity checking in public cloud (ID-PUIC). We give the formal definition, system model, and security model. Then, a concrete ID-PUIC protocol is designed using the bilinear pairings. The proposed ID-PUIC protocol is provably secure based on the hardness of computational Diffie–Hellman problem. Our ID-PUIC protocol is also efficient and flexible. Based on the original client's authorization, the proposed ID-PUIC protocol can realize private remote data integrity checking, delegated remote data integrity checking, and public remote data integrity checking.

Index Terms— *Cloud computing, identity-based cryptography, proxy public key cryptography, remote data integrity checking.*

I. INTRODUCTION

Along with the rapid development of computing and communication technique, a great deal of data is generated. These massive data needs more strong computation resource and greater storage space. Over the last years, cloud computing satisfies the application requirements and grows very quickly. Essentially, it takes the data processing as a service, such as storage, computing, data security, *etc*. By using the public cloud platform, the clients are relieved of the burden for storage management, universal data access with independent geographical locations, *etc*. Thus, more and more clients would like to store and process their data by using the remote cloud computing system.

In public cloud computing, the clients store their massive data in the remote public cloud servers. Since the stored data is outside of the control of the clients, it entails the security risks in terms of confidentiality, integrity and availability of data and service. Remote data integrity checking is a primitive which can be used to convince the cloud clients that their data are kept intact. In some special cases, the data owner may be restricted to access the public cloud server, the data owner will delegate the task of data processing and uploading to the third party, for example the proxy. On the other side, the remote data integrity checking protocol must be efficient in order to make it suitable for capacity-limited end devices. Thus, based on identity-based public cryptography and proxy public key cryptography, we will study ID-PUIC protocol.

A. Motivation

In public cloud environment, most clients upload their data to *PCS* and check their

remote data's integrity by Internet. When the client is an individual manager, some practical problems will happen. If the manager is suspected of being involved into the commercial fraud, he will be taken away by the police. During the period of investigation, the manager will be restricted to access the network in order to guard against collusion. But, the manager's legal business will go on during the period of investigation. When a large of data is generated, who can help him process these data? If these data cannot be processed just in time, the manager will face the lose of economic interest. In order to prevent the case happening, the manager has to delegate the proxy to process its data, for example, his secretary. But, the manager will not hope others have the ability to perform the remote data integrity checking. Public checking will incur some danger of leaking the privacy. For example, the stored data volume can be detected by the malicious verifiers. When the uploaded data volume is confidential, private remote data integrity checking is necessary. Although the secretary has the ability to process and upload the data for the manager, he still cannot check the manager's remote data integrity unless he is delegated by the manager. We call the secretary as the proxy of the manager.

In PKI (public key infrastructure), remote data integrity checking protocol will perform the certificate management. When the manager delegates some entities to perform the remote data integrity checking, it will incur considerable overheads since the verifier will check the certificate when it checks the remote data integrity. In PKI, the considerable overheads come from the heavy certificate verification, certificates generation, delivery, revocation, renewals, *etc*. In public cloud computing, the end devices may have low computation capacity, such as mobile phone, ipad, *etc*. Identity-based public key cryptography can eliminate the complicated certificate management. In order to increase the efficiency, identity-based proxy-oriented data uploading and remote data integrity checking is more attractive. Thus, it will be very necessary to study the ID-PUIC protocol.

B. Related Work

There exist many different security problems in the cloud computing. This paper is based on the research results of proxy cryptography, identity-based public key cryptography and remote data integrity checking in public cloud. In some cases, the cryptographic operation will be delegated to the third party, for example proxy. Thus, we have to use the proxy cryptography. Proxy cryptography is a very important cryptography primitive. In 1996, Mambo *et al.* proposed the notion of the proxy cryptosystem [3]. When the bilinear pairings are brought into the identity-based cryptography, identity-based cryptography becomes efficient and practical. Since identity-based cryptography becomes more efficient because it avoids of the certificate management, more and more experts are apt to study identity-based proxy cryptography. In 2013, Yoon *et al.* proposed an ID-based proxy signature scheme with message recovery. Chen *et al.* proposed a proxy signature scheme and a threshold proxy signature scheme from the Weil pairing.

In public cloud, remote data integrity checking is an important security problem. Since the clients' massive data is outside of their control, the clients' data may be corrupted by the malicious cloud server regardless of intentionally or unintentionally. In order to address the novel security problem, some efficient models are presented. In 2007, Ateniese *et al.* proposed provable data

possession (PDP) paradigm. In PDP model, the checker can check the remote data integrity without retrieving or downloading the whole data. PDP is a probabilistic proof of remote data integrity checking by sampling random set of blocks from the public cloud server, which drastically reduces I/O costs. The checker can perform the remote data integrity checking by maintaining small metadata. After that, some dynamic PDP model and protocols are designed.

From the role of the remote data integrity checker, all the remote data integrity checking protocols are classified into two categories: private remote data integrity checking and public remote data integrity checking. In the response checking phase of private remote data integrity checking, some private information is indispensable. On the contrary, private information is not required in the response checking of public remote data integrity checking. Specially, when the private information is delegated to the third party, the third party can also perform the remote data integrity checking. In this case, it is also called delegated checking.

C. Contributions

In public cloud, this paper focuses on the identity-based proxy-oriented data uploading and remote data integrity checking. By using identity-based public key cryptology, our proposed ID-PUIC protocol is efficient since the certificate management is eliminated. ID-PUIC is a novel proxy-oriented data uploading and remote data integrity checking model in public cloud. We give the formal system model and security model for ID-PUIC protocol. Then, based on the bilinear pairings, we designed the first concrete ID-PUIC protocol. In the random oracle model, our designed ID-PUIC protocol is provably secure. Based on the original client's authorization, our protocol can realize private checking, delegated checking and public checking.

II. SYSTEM MODEL AND SECURITY MODEL OF ID-PUIC

In this section, we give the system model and security model of ID-PUIC protocol. An ID-PUIC protocol consists of four different entities which are described below:

1) *Original Client:* an entity, which has massive data to be uploaded to *PCS* by the delegated proxy, can perform the remote data integrity checking.
2) *PCS (Public Cloud Server):* an entity, which is managed by cloud service provider, has significant storage space and computation resource to maintain the clients' data.
3) *Proxy:* an entity, which is authorized to process the *Original Client*'s data and upload them, is selected and authorized by *Original Client*. When *Proxy* satisfies the warrant m_ω which is signed and issued by *Original-Client*, it can process and upload the original client's data; otherwise, it cannot perform the procedure.
4) *KGC (Key Generation Center):* an entity, when receiving an identity, it generates the private key which corresponds to the received identity.

In our proposed ID-PUIC protocol, *Original Client* will interact with *PCS* to check the remote data integrity. Thus, we give the definition of interactive proof system. Then, we give the formal definition and security model of ID-PUIC protocol.

Definition 1 (ID-PUIC): An ID-PUIC protocol is a collec-tion of four phases (*Setup*, *Extract*, *Proxy-key Generation*, *TagGen*) and an interactive proof system (*Proof*). The detailed phases are described below.

1) *Setup*: When the security parameter k is input, the algorithm outputs the system public parameters and the master secret key. The system public parameters are made public and the master secret key *msk* is made confidential by KGC.
2) *Extract*: When the system public parameters, the master secret key *msk*, and an identity *ID* are input, KGC outputs the private key sk_{ID} which corresponds to the identity *ID*.
3) *Proxy-Key Generation*: *Original Client* generates the warrant m_ω and signs m_ω . Then, it sends the warrant-signature pair to the proxy. Upon receiving the warrant-signature pair from *Original Client*, the proxy generates the proxy-key by using its own private key.
4) *TagGen*: Input the file block F_i and the proxy-key, the proxy generates the corresponding tag T_i . Then, it uploads the block-tag pair to *PCS*.
5) *Proof*: It is an interactive proof system between *PCS* and *Original Client*. At the end of the interactive proof protocol, *Original Client* outputs a bit {0|1} denoting "success" or "failure".

A practical ID-PUIC protocol must be efficient and provably secure. Based on the communication and computation overheads, efficiency analysis can be given. On the other hand, a secure ID-PUIC protocol must satisfy the following security requirements:

1) *Original Client* can perform the ID-PUIC protocol without the local copy of the file(s) to be checked.
2) Oly if the proxy is authorized, *i.e.*, it satisfies the warrant m_ω , the proxy can process the files and upload the block-tag pairs on behalf of *Original Client*.
3) *Original Client* cannot counterfeit the proxy to generate block-tag pairs, *i.e.*, the proxy-protection property is satisfied.
4) If some challenged block-tag pairs are modified or lost, *PCS*'s response cannot pass *Original Client*'s integrity checking.

To capture the above security requirements, we formalize the security definition of an ID-PUIC protocol. First, we give the formal definition of proxy-protection.

Definition 2 (Proxy-Protection): An ID-PUIC protocol satisfies the property of proxy-protection if for the probabilistic polynomial time adversary A_1, the probability that A_1 wins the ID-PUIC game-1 is negligible. The ID-PUIC game-1 between A_1 and the challenger C_1 is given below:

1) *Setup*: The challenger C_1 runs *Set u p* and gets the system public parameters and master secret key. By running *Extract*, C_1 gets *Original Client* ID_o's private key sk_{ID_o} and the proxy ID_p 's private key sk_{ID_p} . It sends the public parameters and *Original Client* ID_o's private key sk_{ID_o} to A_1 while it keeps confidential the master secret key *msk* and the proxy ID_p 's private key sk_{ID_p} .
   - *Oracle queries*: A_1 adaptively queries the oracles *Extract, Hash, Proxy-key Generation, TagGen* to C_1 below:
   - *Extract* queries. A_1 queries the entity *ID*'s private key to C_1. For the identity *ID*, C_1 runs *Extract*
   - *Hash* queries. A_1 queries *hash* oracle to C_1 adaptively. C_1 responds A_1 the *hash* values.
   - *Proxy-key Generation* queries. A_1 sends (ID_o, ID_p) to C_1 and queries the proxy-key where the original client is ID_o and the proxy is ID_p . Denote S as the set which is composed of all the queried original client identity and

proxy identity pairs. The restriction is that $(ID_o, ID_p) \in S$.

- *TagGen* queries. A_1 makes block-tag pair queries adaptively. For the block F_i, C_1 computes its tag T_i and responds A_1 with T_i.

At the end of game-1, A_1 outputs the forged block-tag pair (F, T) with non-negligible probability, where F has not been queried to *TagGen* oracle. If (F, T) is valid block-tag pair, then A_1 wins the above game with non-negligible probability.

Definition 3 states that, if some challenged blocks have been modified or deleted, the malicious *PCS* cannot generate a valid remote data integrity proof. On the other hand, a practical ID-PUIC protocol also needs to convince the client that all of his outsourced data is kept integer with a high probability. The following definition states the security requirement.

Table I Notations and Descriptions

Notations	Descriptions
$\mathcal{G}_1$	Cyclic multiplicative group with order q
$\mathcal{G}_2$	Cyclic multiplicative group with order q
$\mathcal{Z}_q^*$	$\{1, 2, \cdots, q-1\}$
g	A generator of $\mathcal{G}_1$
H, h	Two cryptographic hash functions
f	Pseudo-random function
π	Pseudo-random permutation
(x, Y)	Master secret/public key pair
(ID_o, sk_{ID_o})	Original client's identity-private key pair where $sk_{ID_o} = (R_o, \sigma_o)$
(ID_p, sk_{ID_p})	Proxy's identity-private key pair where $sk_{ID_p} = (R_p, \sigma_p)$
σ	The generated proxy-key
n	The block number
$F = (F_1', \cdots, F_n')$	The stored file F is split into n blocks
PCS	Public cloud server
O	Original client
u	The public parameter which is picked by the original client
(N_i, i)	N_i denotes the i-th block F_i''s name and other properties
(F_i, T_i)	The i-th block-tag pair
v_i	The permutated index $v_i = \pi_{k_1}(i)$ of i
$\theta = (F, T)$,	PCS's response
C_p,	Time cost of bilinear pairings on $\mathcal{G}_1$
C_e,	Time cost of exponentiation on $\mathcal{G}_1$
C_m,	Time cost of multiplication on $\mathcal{G}_1$
m_ω,	The warrant which is generated and signed by the original client

Definition 4 [(ρ, δ) Security]: An ID-PUIC protocol satisfies the security (ρ, δ) if *PCS* corrupted ρ fraction of the whole blocks, the probability that the corrupted blocks are detected is at least δ.

The notations throughout this paper are listed in Table I.

III. THE PROPOSED ID-PUIC PROTOCOL

In this section, we propose an efficient ID-PUIC protocol for secure data uploading and storage service in public clouds. Bilinear pairings technique makes identity-based cryptography practical. Our protocol is built on the bilinear pairings. We first review the bilinear pairings. Then, the concrete ID-PUIC protocol is designed from the bilinear pairings. At last, based on the computation cost and communication cost, we give the performance analysis from two aspects: theoretical analysis and prototype implementation.

A. Bilinear Pairings

Denote G_1 and G_2 as two cyclic multiplicative groups who have the same prime order q. Let Z_q^* denote the multiplicative group of the field F_q. Bilinear pairings is a bilinear map $e : G_1 \times G_1 \rightarrow G_2$ [32] which satisfies the properties below:

1) Bilinearity: $\forall g_1, g_2, g_3 \in G_1$ and $a, b \in Z_q^*$,

$e(g_1, g_2g_3) = e(g_2g_3, g_1) = e(g_2, g_1)e(g_3, g_1)$

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

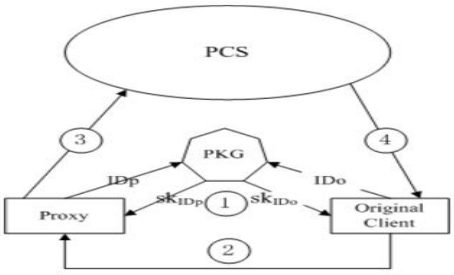

Fig. 1 Architecture of our ID-DPDP protocol

2) Non-degeneracy: $\exists g_4, g_5 \in G_1$ such that $e(g_4, g_5) = {}^1G_2$.

3) Computability: $\forall g_6, g_7 \in G_1$, there is an efficient algorithm to compute $e(g_6, g_7)$.

The concrete bilinear pairings e can be constructed by using the modified Weil [33] or Tate pairings [34] on elliptic curves. Our ID-PUIC protocol construction takes use of the easiness of DDH (Decisional DiffieHellman) problem while its security is based on the hardness of CDH (Computational DiffieHellman) problem. Let g be a generator of G_1. CDH problem and DDH problem are defined below.

Definition 5 (CDH Problem on G_1): Given the triple $(g, g^a, g^b) \in {G_1}^3$ where $a, b \in Z_q$ are unknown, compute $g^{ab} \in G_1$.

Definition 6 (DDH Problem on G_1): Given the quadruple $(g, g^a, g^b, g^{\hat{}}) \in {G_1}^4$ where $a, b \in Z_q^*$ are unknown, decide whether or not the formula $g^{ab} = g^{\hat{}}$ holds.

In the paper, we choose the group G_1 which satisfies the condition that CDH problem is difficult but DDH problem is easy. On the group G_1, DDH problem is easy by using the bilinear pairings. (G_1, G_2) are also called GDH (Gap Diffie-Hellman) groups. On the groups G_1 and G_2, the basic requirement is that the DLP (Discrete Logarithm Problem) is difficult. Let g_2 be a generator of G_2. It is given below:

B. Our Concrete ID-PUIC Protocol

This concrete ID-PUIC protocol comprises four procedures: *Setup*, *Extract*, *Proxy-key generation*, *TagGen*, and *Proof*. In order to show the intuition of our construction, the concrete protocol's architecture is depicted in Figure 1. First, *Setup* is performed and the system parameters are generated. Based on the generated system parameters, the other procedures are performed as Figure 1. It is described below: (1) In the phase *Extract*, when the entity's identity is input, KGC generates the entity's private key. Especially, it can generate the private keys for the client and the proxy. (2) In the phase *Proxy-key generation*, the original client creates the warrant and helps the proxy generate the proxy key. (3) In the phase *TagGen*, when the data block is input, the proxy generates the block's tag and upload block-tag pairs to *PCS*. (4) In the phase *Proof*, the original client *O* interacts with *PCS*. Through the interaction, *O* checks its remote data integrity.

Following the protocol's architecture, we give the concrete construction below. Without loss of generality, suppose that the proxy plans to upload the file F. According to the size of F, we split it into multiple blocks. Suppose that F is split into n blocks, *i.e.*, $F = (F_1, \cdots, F_n)$. F_i denotes the i-th block of F. Let N_i contain the name and attributes of the block F_i . (N_i, i) will be used to create the tag of F_i . The phases are described in detail as the following.

- *Setup*: Let G_1, G_2 be the two groups and e be the bilinear pairings which are given in the section III-A. Both G_1 and G_2 have the same order q. Let g be a generator g of the group G_1. Two cryptographic hash functions are given below:

$H: \{0, 1\}^* \rightarrow Z_q^*$, $h: Z_q^* \times \{0, 1\}^* \rightarrow G_1$

Pick a pseudo-random function f and a pseudo-random permutation π . The two functions f and π are defined below:

$$f: Z_q^* \times \{1, 2, \cdots, n\} \rightarrow Z_q^*$$

$$\pi: Z_q^* \times \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, n\}$$

KGC generates its master secret key x where $x \in Z_q^*$. Then, it computes $Y = g^x$.

The parameters $\{G_1, G_2, e, q, g, Y, H, h, f, \pi\}$ are made public. The mas-ter secret key x is kept confidential by KGC.

- *Extract*: Input the original client's identity $I D_o$, KGC picks a random $r_o \in Z_q^*$ and computes (R_o, σ_o) below:

  $R_o = g^{ro}$, $\sigma_o = r_o + x\, H(I D_o, R_o)$ mod q

  Then, KGC sends $sk_{I\,Do} = (R_o, \sigma_o)$ to the original client by the secure channel. Let $sk_{I\,Do}$ be the original client's private key. The original client verifies $sk_{I\,Do}$'s correctness by verifying the following equation.

  $$g\sigma o = RoY^{H}(^{I}Do, ^{R}o) \quad (1)$$

  If the formula (1) holds, the original client $I D_o$ accepts $sk_{I\,Do}$ as its private key; otherwise, $I D_o$ rejects it and requests its private key by using *Extract* again. Similarly, input the proxy's identity $I D_p$, the proxy $I D_p$ can also get its private key $sk_{I\,D\,p} = (R_p , \sigma_p)$.
- *Proxy-key generation*: In order to generate the proxy key, the original client $I D_o$ will interact with the proxy $I D_p$ below:
  1) $I D_o$ creates the warrant m_ω according to its require-ments. The proxy $I D_p$ cannot process and upload the original client $I D_o$'s data unless it satisfies m_ω . $I D_o$ picks a random $r_1 \in Z_q^*$ and computes m_ω 's signature below:

     $R_1 = g^{r1}$, $\sigma_1 = r_1 + \sigma_o\, H(m_\omega, R_1)$ mod q

     $I D_o$ sends the warrant-signature pair $(m_\omega , R_1, \sigma_1)$ and R_o to $I D_p$ and *PCS*.

  2) $I D_p$ checks the validity of $(m_\omega , R_1, \sigma_1, R_o)$ by verifying whether or not the following equation holds.

     $$g^{\sigma 1}_{R1} = R1(RoY^{H(I D o, R o)})^{H(m\omega ,}$$

     If the verification is unsuccessful, the proxy rejects it and informs $I D_o$; otherwise, it computes the proxy secret key:

     $\sigma = \sigma_1 + \sigma_p\, H(m_\omega, R_1)$

     The proxy secret key σ is kept secret by the proxy.

     At the same time, $I D_p$ sends R_p to $I D_o$.
- *TagGen*: When $I D_p$ satisfies the warrant m_ω , $I D_p$ will help $I Do$ process its data. Suppose the original client's ˆplaintext file is F. By using the light-weight symmetric encryption, F is encrypted into the ciphertext F which will be uploaded to *PCS*. Based on the size of F, the proxy $I D_p$ splits F into n blocks, *i.e.*, $F = (F_1, \cdots, F_n)$. F_i denotes the i-th block of F and $F_i \in Z_q^*$. N_i contains the i-th block F_i 's name and its properties. The proxy calculates $u = h(n + 1, I D_0)$. Then, for $1 \leq i \leq n$, the proxy performs the following procedures step by step:
  1) The proxy computes $T_i = (h(i, N_i)u^{F}i)^{\sigma}$ by using the proxy key σ ;
  2) The proxy outputs the block F_i 's tag T_i .

  At last, the proxy gets all the block-tag pairs $\{(F_i , T_i), 1 \leq i \leq n\}$ and uploads them to *PCS*.

  When *PCS* receives m_ω 's signature $(m_\omega , R_1, \sigma_1)$ and R_o,

  PCS checks whether $I D_p$ satisfies m_ω . If it holds, *PCS* accepts and stores them; otherwise, *PCS* refuses to accept them.
- *Proof (PCS, O)*: This is a 2-move interactive protocol between *PCS* and the original client *O*. If *O* authorizes the remote data integrity checking task to some verifier, it sends (R_o, R_p , R_1) to the authorized verifier. The authorized verifier

may be the third auditor or *O*'s proxy. Since *O* has *(R_o, R_p , R_1), O* can performs the interactive protocol *Proof* as the verifier. When the verifier is *O*, the interaction protocol *Proof* is given below.

1) Challenge (*O* → *PCS*): *O* generates the challenge *chal = (c, k_1, k_2).* In *chal*, *c* is the challenged block number which is determined by *O* and k_1, k_2 are randomly picked from Z_q^*. Then, it sends the challenge *chal* to *PC S*;

2) Response (*O* ← *PCS*): Upon receiving :– *O*'s chal-lenge *chal = (c, k_1, k_2), PCS* performs the follow-ing procedures:

a) Search for the two-tuples *(O, k_1)* in the table T_{chal} . If *(O, k_1)* ∈ T_{chal} , *PCS* refuses and informs the requester *O*; otherwise, it goes on;

b) For $1 \le i \le c$, *PCS* computes $v_i = \pi_{k1}$ *(i)*, $a_i = f_{k2}$ *(i)* by using k_1, k_2, When *O* outputs "failure", it will perform the same challenge many times. If the responses still cannot pass the verification, *O* will inform the *PCS* manager this situation. *PCS* manager will censor *O*'s stored data and retrieve the lost data from the offline backup. If *PCS* manager fails, *O* and *PCS* manager will have to evaluate the loss and discuss the reparation according to the loss severity.

C. Performance Analysis

First, we give the computation and communication overhead of our proposed ID-PUIC protocol. At the same time, we implement the prototype of our ID-PUIC protocol and evaluate its time cost. Then, we give the flexibility of remote data integrity checking in the phase *Proof* of our ID-PUIC protocol. At last, we compare our ID-PUIC protocol with the other up-to-date remote data integrity checking protocols.

1) Computation: Let the uploaded block number be *n*. Let the challenge be *chal = (c, k_1, k_2)* where $1 \le c \le n$, $k_1, k_2 \in Z_q^*$. Based on the different phases, we give the computation overhead. On the group G_1, bilinear pairings, exponentiation, and multiplication contribute most computation cost. Compared with them, the other operations are faster, such as, hash function *h*, the operations on Z_q^* and G_2, *etc*. The hash function *H* can be done once for all. Thus, we only consider bilinear pairings, exponentiation, and multiplication on G_1. For the proxy, the computation overhead mainly comes from the phase *T agGen*. In the phase *TagGen*, the proxy performs 2*n* exponentiation, *n* multiplication on the group G_1, and *n* hash function *h*. In the phase *Proof*, the original client *O* generates the challenge *chal* and *PCS* responds to *chal*. In the interactive proof *Proof*, *PCS* performs *c* exponentiation and *c* − 1 multiplication on the group G_1 and sends the response θ to the checker *O*. In order to check the response θ 's validity, *O* performs c+3 exponentiation, 2 pairings, 3+*c* multiplication on the group G_1 and *c* hash function *h*. In order to show our protocol's practical computation overhead, we have simulated the proposed ID-PUIC protocol by using C programming language with GMP Library (GMP-5.0.5), and PBC Library (pbc-0.5.13).

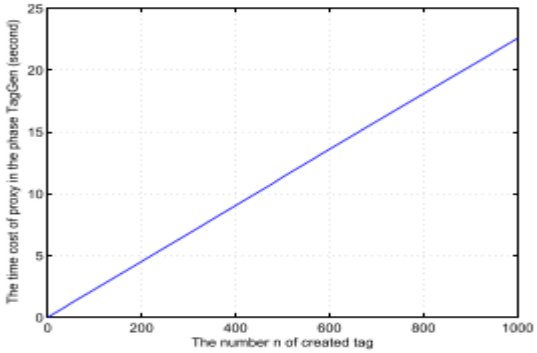

Fig. 2. Proxy's time cost in TagGen (second)

In the simulation, *PCS* works on DELL PowerEdge R420 Server with the following settings:

- CPU: Intel R Xeon R processor E5-2400 and E5-2400 v2 product families
- Physical Memory: 8GB DDR3 1600MHz
- OS: Ubuntu 13.04 Linux 3.8.0-19-generic SMP i686

The proxy and the original client works on PC Laptop with the following settings:

- CPU: CPU I PDC E6700 3.2GHz
- Physical Memory: DDR3 2G 1600MHz
- OS: Windows 7

In the simulation, we choose type A pairing parameters, where the group order is 160 bits, and the order of the base field is 512bit. Type A pairing parameters provide a competitive security level with 1024-bit RSA. In *TagGen*, when one tag is created, proxy's time cost is 0.022472s. When one thousand tags are created, proxy's time cost is 22.567269s. In order to show the time cost, we give Figure 2. Figure 2 depicts proxy's computation cost in *TagGen*. X-label denotes the created tag number n and Y-label denotes the time cost (second) in order to create n tags. Proxy's time cost increases almost linearly with the created tag number n. In *Proof*, the original client interacts with *PCS* and performs the remote data integrity checking. In *Proof*, when the challenged block number is 20, the original client's time cost is 0.567881s. When the challenged block number is 100, the original client's time cost is 2.356659s. In order to show the time cost, we give Figure 3. Figure 3 depicts original client's computation time cost in *Proof*. X-label denotes the challenged block number c and Y-label denotes the time cost (second) in order to check *PCS*'s response θ. At the same time, we consider the computation time cost of *PCS*. In *Proof*, when the challenged block number is 20, the *PCS*'s time cost is 1.611138s. When the challenged block number is 100, the *PCS*'s time cost is 7.993312s. In order to show the time cost, *PCS*'s time cost figure is added to Figure 3. Figure 3 also depicts *PCS*'s computation time cost in *Proof*. X-label denotes the challenged block number c and Y-label denotes *PCS*'s time cost (second).

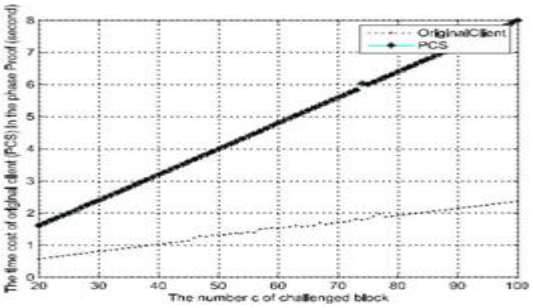

Fig. 3. PCS and Original Client's time cost in Proof (second)

2) Communication: National Bureau of Standards and ANSI X9 have determined the shortest key length requirements: RSA and DSA is 1024 bits, ECC is 160 bits [35]. According to the above standard, we analyze our ID-PUIC protocol's communication cost. After the data processing, the block-tag pairs are uploaded to *PCS* once and for all. Thus, we only consider the communication cost which is incurred in the remote data integrity checking. In *Proof*, the communication cost comprises the challenge *chal* and response θ. The original client will interact with *PCS* periodically in the phase *Proof*. Suppose there are n message blocks are stored in the *PCS*.

In order to finish one round interaction, the original client will create the challenge $chal = (c, k_1, k_2)$ and send *chal* to *PCS*. The whole communication cost is $\log_2 n + 2 \log_2 q = 320 + \log_2 n$ bits. In order to respond the challenge *chal*, *PCS* creates response $\theta = (\quad, \quad)$ θ 's bit length is the $F \quad T$.

$160 + 1|G_1| = 160 + 2 * 512 = 1184$ bits. Thus, for one round interaction of *Proof*, the whole communication cost is $320 + \log_2 n + 1184 = 1504 + \log_2 n$ bits.

3) *Private Checking, Delegated Checking and Public Checking:* Our proposed ID-PUIC protocol satisfies the private checking, delegated checking and public checking. In the remote data integrity checking procedure, R_1, R_o, R_p are indispensable. Thus, the procedure can only be performed by the entity who has R_1, R_o, R_p . In general, since R_1, R_o, R_p are kept secret by the original client, our protocol can only be performed by the original client. Thus, it is private checking. On some cases, the original client has no ability to check its remote data integrity, such as, he is taking a vacation or in prison or in battle field, *etc*. Thus, it will delegate the third party to perform the ID-PUIC protocol. It can be the third auditor or the proxy or other entities. The original client sends R_1, R_o, R_p to the delegated third party. The delegated third party has the ability to perform the ID-PUIC protocol. Thus, it has the property of delegated checking. On the other hand, if the original client makes R_1, R_o, R_p public, any entity has the ability to perform the ID-PUIC protocol. Thus, our protocol has also the property of public checking.

Notes: In the checking, the checker must have R_1, R_o, R_p . R_o, R_p are the part of original client's private key and the proxy's private key respectively. Their publicity cannot leak their the other part of private key, *i.e.*, σ_o, σ_p cannot be leaked. The private key extraction phase *Extract* is actually a modified ElGamal signature scheme which is existentially unforgeable. For the identity *ID*, the extracted private key $R, \sigma)$ is a signature of *ID*. Since ElGamal signature is existentially unforgeable, the private key part σ will keep secret even if R is made public. On the other hand, R_1 is generated by the original client in order to create the signature on the warrant m_ω . Thus, R_1 is also known to the original client.

4) *Comparison:* In order show our ID-PUIC protocol's superiority, we compare our protocol with the recent two protocols, *i.e.*, Wang's protocol and Zhang *et al.*'s protocol. Since most computation cost comes from the pair-ings, exponentiation and multiplication on the group G_1, the simplified comparison is given in Table II. In the comparison, we denote the time cost of pairings, exponentiation and multiplication as C_p , C_e , C_m , respectively. From the computation comparison, we know that our protocol has almost the same computation cost in the phases *TagGen* and PCS has the same computation cost in the phase *Proof*. For the checker in the phase *Proof*, our protocol has less computation cost than the other two protocols. On the other hand, our protocol can realize the three security properties: proxy data processing and uploading, remote data integrity checking with flexibility and not required certificate management. Flexibility means that our protocol can realize the private checking, delegated checking and public checking according to the original client's authorization.

5) *Hybrid Cloud:* Our contributions are also suitable for the scenario of hybrid clouds, where the proxy can be treated as the private cloud of the original client. In the scenario, the private cloud gets its private/public key pairs. The private cloud gets the proxy-key and the authorization of the original client through the interaction between the original client and its private cloud. When the original client needs its private cloud perform the data uploading task, it informs its private cloud. Upon receiving the original client's instruction, the

private cloud will interact with the public cloud and finish the data uploading task.

IV. SECURITY ANALYSIS

The security of our ID-PUIC protocol mainly consists of the following parts: correctness, proxy-protection and unforgeability. The correctness has been shown in the subsection III-B. In the following paragraph, we study the proxy-protection and unforgeability. Proxy-protection means that the original client cannot pass himself off as the proxy to create the tags. Unforgeability means that when some challenged blocks are modified or deleted, *PCS* cannot send the valid response which can pass the integrity checking.

In order to formally prove our protocol, we give the definition of *(t,)*-security below:

Theorem 1 (Unforgeability): The proposed ID-PUIC proto-col is existentially unforgeable in the random oracle model if CDH problem on G_1 is hard. Since the proof process is almost the same as Shacham-Waters's protocol [20], we only give the differences. In Shacham-Waters's protocol, u is randomly picked from G_1. In our ID-PUIC protocol, u is calculated by using the hash function h. In the random oracle model, h's output value is indistinguishable from a random value nn the group G_1. In the phase *TagGen*, the proxy-key σ is used in ID-PUIC protocol while the data owner's secret key a is used in Shacham-Waters's protocol [20]. For *PCS*, σ and a has the same function to generate the block tags. When *PCS* is

Fig. 4 The probability curve P_X of detecting the corruption.

dishonest, since Shacham-Waters's protocol is existentially unforgeable in random oracle model, our proposed ID-PUIC protocol is also existentially unforgeable in the random oracle model. The detailed proof process is omitted since it is very similar to Shacham-Waters's protocol.

V. CONCLUSION

Motivated by the application needs, this paper proposes the novel security concept of ID-PUIC in public cloud. The paper formalizes ID-PUIC's system model and security model. Then, the first concrete ID-PUIC protocol is designed by using the bilinear pairings technique. The concrete ID-PUIC protocol is provably secure and efficient by using the formal security proof and efficiency analysis. On the other hand, the proposed ID-PUIC protocol can also realize private remote data integrity checking, delegated remote data integrity checking and public remote data integrity checking based on the original client's authorization.

REFERENCES

[1] *Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing," IEICE Trans. Commun., vol. E98-B, no. 1, pp. 190–200, 2015.*

[2] *Y. Ren, J. Shen, J. Wang, J. Han, and S. Lee, "Mutual verifiable provable data auditing in public cloud storage," J. Internet Technol., vol. 16, no. 2, pp.317–323, 2015.*

[3] *M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegating signing operation," in Proc. CCS, 1996, pp. 48–57.*

[4] *E.-J. Yoon, Y. Choi, and C. Kim, "New ID-based proxy signature scheme with message recovery," in Grid and Pervasive Computing (Lecture Notes in Computer Science), vol. 7861. Berlin, Germany: Springer-Verlag, 2013, pp. 945–951.*

[5] *B.-C. Chen and H.-T. Yeh, "Secure proxy signature schemes from the weil pairing," J. Supercomput., vol. 65, no. 2, pp. 496–506, 2013.*

[6] *X. Liu, J. Ma, J. Xiong, T. Zhang, and Q. Li, "Personal health records integrity verification*

using attribute based proxy signature in cloud computing," in Internet and Distributed Computing Systems (Lecture Notes in Computer Science), vol. 8223. Berlin, Germany: Springer-Verlag, 2013, pp. 238–251.

[7] H. Guo, Z. Zhang, and J. Zhang, "Proxy re-encryption with unforgeable re-encryption keys," in Cryptology and Network Security (Lecture Notes in Computer Science), vol. 8813. Berlin, Germany: Springer-Verlag, 2014, pp. 20–33.

[8] E. Kirshanova, "Proxy re-encryption from lattices," in Public-Key Cryptography (Lecture Notes in Computer Science), vol. 8383. Berlin, Germany: Springer-Verlag, 2014, pp. 77–94.

[9] P. Xu, H. Chen, D. Zou, and H. Jin, "Fine-grained and heterogeneous proxy re-encryption for secure cloud storage," Chin. Sci. Bull., vol. 59, no. 32, pp. 4201–4209, 2014.