Secure Lightweight Stream Data Outsourcing for Internet of Things

Su Peng, Liang Zhao, Ahmed Y. Al-Dubai, *Senior Member, IEEE,* Albert Y. Zomaya, *Fellow, IEEE,* Jia Hu, Geyong Min, and Qiang Wang

***Abstract*—The epoch of the Internet of Things (IoT) has come by enabling almost everything to gather and share electronic information. Considering the unreliable factors of public IoT, how to outsource huge amounts of indispensable stream data generated by the nodes to the remote storage efficiently and securely is one of the most challenging issues. In this paper, we propose a secure lightweight stream data outsourcing framework for IoT based on identity and blockchain. Taking advantage of identity-based cryptography and blockchain, for public IoT containing untrusted communication channels, nodes, remote storage, and even verifiers, we introduce a private mobile network and multiple verifiers to ensure that the stream data are stored intact and updated correctly, without the costs and risks brought by the Public Key Infrastructures (PKI). Meanwhile, the framework can also achieve privacy-preserving checking, by revealing no data to the other entities besides the remote storage, even in the blockchains. Our comprehensive analysis and experiments demonstrate that the proposed framework is suitable for lightweight devices and practical for IoT.**

***Index Terms*—Internet of Things, remote data integrity checking, lightweight, identity-based cryptography, blockchain**

I. Introduction

THE Internet of Things (IoT) is considered as one of the most revolutionary technology of the 21st century. IoT is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment [1]. These objects can be heart monitor implants, farm animals with biochip transponders, and automobiles. All the objects and devices in IoT can transfer information without requiring human-to-human or human-to-computer interaction. Derived from the ubiquitous Internet technology, IoT is now experiencing an exponential growth, to become the next major step in delivering the Internet's promise of making the world a connected place. However, along with the dramatic development, some challenges of IoT arise, for instance, outsourcing the data securely [2].

This paper is supported in part by the Key Projects of Liaoning Provincial Department of Education Science Foundation under Grant L201702, the National Natural Science Foundation of China under Grant 61701322, the Young and Middle-aged Science and Technology Innovation Talent Support Plan of Shenyang under Grant RC190026, the Natural Science Foundation of Liaoning Province under Grant 2020-MS-237, the Liaoning Provincial Department of Education Science Foundation under Grants JYT19052, JYT2020109, and the Research Start-up Fund for Fresh Talent of SAU under Grant 20YB02.

Su Peng (supeng@stumail.neu.edu.cn) and Liang Zhao (lzhao@sau.edu.cn) are with Shenyang Aerospace University, China. Liang Zhao is the corresponding author.

Ahmed Y. Al-Dubai (a.al-dubai@napier.ac.uk) is with Edinburgh Napier University, United Kingdom.

Albert Y. Zomaya (albert.zomaya@sydney.edu.au) is with University of Sydney, Australia.

Jia Hu (J.Hu@exeter.ac.uk) and Geyong Min (g.min@exeter.ac.uk) are with University of Exeter, United Kingdom.

Qiang Wang (wangq0427@gmail.com) is with Northeastern University, China.

A. Secure Stream Data Outsourcing in IoT

Tremendous amount of IoT devices produce a great amount of data every day, which play a crucial role in the modern industries, such as smart manufacturing, healthcare, and insurance. Besides their immeasurable commercial value, these massive data are indispensable for the other next generation information technologies. For instance, training data are considered as the blood of deep learning. Furthermore, stream files collected by sensors, microphones, and cameras account for a large proportion of these data. However, most of the nodes in IoT are low-powered devices. For the economic and energy-saving purposes, the storage on these devices is usually quite limited. Accordingly, their owners (e.g., corporations, institutions, and individuals) may wish to outsource the collected stream data to the remote storage (RS), such as the third-party cloud or edge computing servers for further processing, in which some security issues emerge. Although the RS may apply general protections for the data, such means rely solely on their reputations and skills, while the original data owners have no control. Catastrophic data loss may occur because of some artificial (e.g., administrator errors, malicious insiders, and hackers' invasions) or non-artificial (e.g., media damages, power failures, and natural disasters) factors. For instance, on August 31st, 2019, an Amazon AWS US-EAST-1 data center in North Virginia experienced a power failure. After the power was restored, some EC2 instances and EBS volumes incurred hardware damage and the data stored on them were no longer recoverable [3]. What is worse, the data hosted by the edge computing servers are more vulnerable than those stored in the data centers because of the poor environments.

B. Overview and Limitations of Current Solutions

As mentioned above, it is critical for data owners to confirm that their data are well maintained on the RS without downloading them, namely, Remote Data Integrity Checking (RDIC). The first solution concerning this problem was proposed by Ateniese *et al.* [4] in 2007, which is called Provable Data Possession (PDP). In PDP, the client splits the entire data file F into blocks, i.e., $F = (m_1, m_2, \ldots, m_n)$, creates a Homomorphic Verifiable Tag (HVT) t_i for each block m_i, then

uploads each m_i and t_i to the RS. Later, the client (or the third-party verifier) chooses some indices randomly and asks the RS to prove that these blocks are still intact, namely, a challenge. Accordingly, the RS forms a proof by generating an aggregated block M_i and an aggregated HVT T_i from the chosen blocks $\{m_i\}$ and the corresponding HVTs $\{t_i\}$, respectively. After receiving the proof, the client (or the verifier) substitutes M_i and T_i into an equation. If it holds, then the proof is valid, which implies the integrity of all the challenged blocks. The communication overhead of PDP is constant, independent to the size of F, because M_i and T_i are the same sizes as m_i and t_i, respectively. Meanwhile, it introduces a third-party verifier to release the client from the complicated integrity checking task. Following this direction, many RDIC schemes have been proposed in different aspects, such as data recovery [5], [6], user revocation [7]–[9], key-exposure resistance [10], [11]. Especially, some identity-based RDIC schemes have been proposed in recent years [9], [12]–[16], which realizes third-party integrity checking without the Public Key Infrastructures (PKI). However, most of the current solutions have some shortcomings while applying to IoT, which can be stated as follows:

1) Computational Complexity: For a secure RDIC scheme, the smallest data unit (block or sector) must match to a unique element in a particular group, such as a multiplicative group in a finite field, or an elliptic curve group, whose order must be at least 2048 bits or 160 bits. However, this size is too small for a large file. For instance, a file of 1 GB must be split into at least 4,194,304 units when the unit size is 2048 bits. The HVT-generation algorithms in most of the current solutions carry out at least one large-number exponentiation for each unit, which makes them very slow, especially for low-powered IoT nodes holding large files. Although we can decrease the unit number by enlarging the group size, the computational complexity for generating each HVT increases exponentially, hence it only makes the matter worse. Although several schemes adopt online/offline signatures to accelerate this part [17], [18], they require the nodes to pre-store many offline tags (even more than the online ones), which are still unaffordable for devices with limited storage.

2) Over-idealized Security Model: The third-party verifier is assumed to be fully trusted, which may be too ideal. Because RDIC is usually complicated and resource-consuming for IoT nodes, it is reasonable for the data owner to delegate the verification tasks to a public verifier. Hence, the similar security issues as in the RS may also appear. However, in most of the current solutions, the verifiers just send the results back to the data owners without any other testimony. In the worst case, if the verifiers conspire with the malicious RS, the data owners cannot distinguish between the honest and fake results. Even the verifiers are always honest, the checking results delivered directly via the public channels may suffer from the man-in-the-middle attacks, thus are still vulnerable to tampering.

3) Resource-consuming Data Updates: The updates of stream data can be summarized as three types, creating new files, deleting existing files, and appending new data to existing files. The other operations (e.g., modifications and insertions) are unusual, even forbidden. However, most of the existing solutions mainly focus on very fine-grained data updates, including insertions, deletions, and modifications at all the positions of the files in the block-level, which invoke heavy computation and communication overheads by verifying Authenticated Data Structures (ADS) (e.g., authenticated skip lists [19] and Merkle hash trees [6], [20], [21]) for each new block, which is unaffordable for low-powered nodes.

C. Contributions

For resolving these problems mentioned above, we propose a secure lightweight stream data outsourcing framework for IoT based on identity and blockchain, with the following features:

1) Identity-based, Lightweight, and Privacy-preserving RDIC (IDLP-RDIC): We propose an RDIC scheme, namely IDLP-RDIC, which is suitable for resource-limited devices in IoT, because its HVT-generation uses much less large-number exponentiations. More precisely, the computational complexity is nearly $O(n)$, where n denotes the block number, no matter how many sectors each block contains. Meanwhile, taking advantage of the identity-based cryptography, the entities in IDLP-RDIC can be explicitly authenticated without the PKI. Furthermore, the integrity checking in IDLP-RDIC does not reveal any content of the data to the other entities, which prevent the leakage of the owner's trade secrets and privacy.

2) Lightweight and Secure Stream Data Updates: For stream files, our framework supports creation, deletion, and appending in the file-level, which are combined with IDLP-RDIC to force the RS to perform them correctly, without any additional overheads brought by the ADS.

3) Reliable under Untrusted Environments: Our framework is available for public IoT, which may contain untrusted communication channels, nodes, RS, and even verifiers. Taking advantage of the mobile computing, the private keys are distributed only during the registrations, by a Private Mobile Network (PMN) acting as a Public Key Generator (PKG), which shrinks the attack surface greatly. Meanwhile, the integrity checking tasks and data updates are supervised by multiple verifiers and integrated into Identity-based Blockchains (IDBCs), which are auditable and traceable, via the novel consensus mechanisms namely the Proof of Verification (PoV), without the power-consuming and high-latency mining operations.

D. Organizations

The rest of this paper is organized as follows. Section II provides some brief reviews of the technologies from which our framework is built. Section III proposes the IDLP-RDIC scheme. Section IV integrates IDLP-RDIC to a comprehensive framework to realize secure stream data outsourcing for IoT. Section V provides security analysis of the framework. Section VI provides detailed theoretical and experimental evaluations of both IDLP-RDIC and the framework. Section VII concludes this paper. Furthermore, Appendix A provides a detailed security proof of IDLP-RDIC.

II. Preliminaries

The building blocks of our framework include the identity-based cryptography, the bilinear pairings, the computational Diffie-Hellman (CDH) problem, the discrete logarithm (DL) problem, and the blockchain.

A. Identity-based Cryptography

In a traditional public-key cryptography system, such as RSA or ElGamal, the public key usually looks like a bunch of unreadable characters, for it is generated from a random private key using some mathematical operations. To identify the public key for a certain entity, the most common way is to retrieve the corresponding certificate from the PKI. However, PKI brings about some other issues, such as the complicated, or even untrusted management of certificates [22]. To remedy these defects, identity-based cryptography systems are proposed [23], [24], whose public keys are identities (e.g., names, phone numbers, and e-mail addresses), which can be self-authenticated without the certificates. For instance, if Alice signs a message using an identity-based signature, Bob can verify the message using Alice's e-mail address instead of her hard-to-remember public key retrieved from the PKI.

B. Bilinear Pairings

Let G and G_T be two cyclic multiplicative groups with the same prime order q and let g be a generator of G. Let $e : G \times G \to G_T$ be a bilinear map [24] which satisfies the following properties:

1) *Bilinearity.* For any $u, v \in G$ and $a, b \in Z_q$, $e(u^a, v^b) = e(u, v)^{ab}$.
2) *Non-degeneracy.* $e(g, g) \neq 1$.
3) *Computability.* For any $u, v \in G$, $e(u, v)$ can be computed efficiently.

C. Computational Diffie-Hellman (CDH) Problem

Let G be a cyclic multiplicative group with the prime order q and let g be a generator of G. Given $(g, g^a, h) \in G^3$ for randomly chosen $a \in Z_q$ and $h \in G$, the CDH problem on G is to compute $h^a \in G$. The CDH assumption on G is that, for any probabilistic polynomial time algorithm, the advantage in solving the CDH problem on G is negligible.

D. Discrete Logarithm (DL) Problem

Let G be a cyclic multiplicative group with the prime order q and let g be a generator of G. Given $(g, h) \in G^2$ for randomly chosen $h \in G$, the DL problem on G is to compute $a \in Z_q$ satisfies $h = g^a$. The DL assumption on G is that, for any probabilistic polynomial time algorithm, the advantage in solving the DL problem on G is negligible.

E. Blockchain

Blockchain was first introduced by Satoshi Nakamoto [25] as the backbone technology of Bitcoin, while some recent advances can be found in [26]–[28]. Literally, in a blockchain, the data is recorded in *blocks* and *chained* with each other using cryptographic hash functions. All the records in a blockchain are immutable, i.e., no participant can change or tamper with a record after it has been written to a blockchain, since the hash value of each block is contained in the subsequent block in the chain. The records are also verifiable and traceable via consensus mechanisms, such as Proof of Work (PoW) and Proof of Stake (PoS). Blockchain is considered as one significant property of the next-generation Internet, to build a novel and reliable large-scale cooperation mode, which has already been applied to various fields, such as finance, supply chain, copyright, and energy. To realize authenticatable and traceable stream data integrity checking, we set up a consortium blockchain network, in which the nodes are pre-registered verifiers.

III. The Proposed IDLP-RDIC Scheme

In this section, we propose a new construction of lightweight HVT. Furthermore, inspired by the SS-2-IBS transform [29] and the blinded linear combination [30], we build IDLP-RDIC from the proposed HVT. IDLP-RDIC consists of six polynomial-time algorithms which are defined as follows:

1) $Setup(1^\kappa) \to (params, x, mpk, S, s)$. The PKG chooses two cyclic multiplicative groups G, G_T with the same prime order $q > 2^\kappa$ and one bilinear map $e : G \times G \to G_T$. Let g be the generator of G. The PKG also chooses four cryptographic hash functions $H_1 : \{0,1\}^* \to G$, $H_2 : \{0,1\}^* \to G$, $h_1 : Z_q \to Z_q$, $h_2 : G_T \to Z_q$ and one pseudo-random function (PRF) $f : Z_q \times \{0,1\}^* \in Z_q$, sets the public parameters $params = \{G, g, G_T, q, e, H_1, H_2, h_1, h_2, f\}$. Then, the PKG chooses a random number $x \in Z_q$ as the master private key, sets the master public key $mpk = g^x$, chooses a sector size S (in bytes, i.e., $S < (\log_2 q)/8$) and a sector number s, publishes mpk, $params$, S, and s, whereas keeps x secret.
2) $Extract(NID, s, x) \to sk$. The data owner forwards his identity $NID \in \{0,1\}^*$ to the PKG. The PKG chooses two random numbers $\alpha, \beta \in Z_q$, computes $v = g^\alpha, \{u_l = g^{h_1(\beta+l-1)} \mid l = 1, \ldots, s\}, \sigma = H_1(NID \parallel v \parallel u_1 \parallel \ldots \parallel u_s)^x$, sets the private key $sk = (\alpha, \beta, \sigma)$ and forwards it to the data owner. The data owner can verify the correctness of sk by checking whether:
$$\begin{aligned} & e(\sigma, g) \\ = \; & e(H_1(NID \parallel g^\alpha \parallel g^{h_1(\beta)} \parallel \ldots \parallel g^{h_1(\beta+s-1)}), \\ & mpk). \end{aligned} \quad (1)$$
3) $HVT-Gen(NID, \{FID_j\}, \{m_{jk}\}, s, sk) \to (\{t_{jk}\}, E)$. Suppose that each file FID_j has been split into n_j blocks $\{m_{jk} \mid k = 1, \ldots, n_j\}$ of Ss bytes. The data owner further splits each block into s sectors $\{\tilde{m}_{jkl} \mid l = 1, \ldots, s\}$ of S bytes, sets each HVT $t_{jk} = (H_2(NID \parallel FID_j \parallel k) \cdot g^{\sum_{l=1}^{s} h_1(\beta+l-1)\tilde{m}_{jkl}})^\alpha$, sets the extra authentication information $E = (v, u_1, \ldots u_s, \sigma)$, then forwards $(\{m_{jk}\}, \{t_{jk}\}, E)$ to the RS.

4) $Challenge() \rightarrow C$. The verifier selects a set I of block index tuples $\{(j,k)\}$ and a random temporary key $\tau \in Z_q$, then forwards the challenge token $C = (I, \tau)$ to the RS.
5) $ProofGen(C, \{m_{jk}\}, \{t_{jk}\}, E, s) \rightarrow P$. According to $C = (I, \tau)$, the RS selects another random temporary key $\gamma \in Z_q$, computes $\Gamma = e(\prod_{l=1}^{s} u_l, v)^{\gamma}$, $\delta = h_2(\Gamma)$ and $\{a_{jk} = f_\tau(NID \parallel FID_j \parallel k) \mid (j,k) \in I\}$, computes the aggregated block $M = (\tilde{M}_1, \ldots, \tilde{M}_s)$ in which $\tilde{M}_l = \gamma + \delta \sum_{(j,k)\in I} a_{jk} \tilde{m}_{jkl}$ and the aggregated HVT $T = \prod_{(j,k)\in I} {t_{jk}}^{a_{jk}}$, then forwards the proof of integrity $P = (M, T, E, \Gamma)$ to the verifier.
6) $ProofCheck(NID, \{FID_j\}, C, P, s, mpk) \rightarrow \{\text{VALID, INVALID}\}$. According to $P = (M, T, E, \Gamma)$, the verifier first verifies E by checking whether:

$$e(\sigma, g) = e(H_1(NID \parallel v \parallel u_1 \parallel \ldots \parallel u_s), mpk). \quad (2)$$

If it does not hold, the verifier outputs INVALID. Otherwise, the verifier computes $\delta = h_2(\Gamma)$ and $\{a_{jk} = f_\tau(NID \parallel FID_j \parallel k) \mid (j,k) \in I\}$, then outputs VALID or INVALID by checking whether the following equation holds:

$$\begin{aligned} &\Gamma \cdot e(T, g)^{\delta} \\ &= e((\prod\nolimits_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\delta} \\ &\cdot \prod\nolimits_{l=1}^{s} {u_l}^{\tilde{M}_l}, v). \end{aligned} \quad (3)$$

IV. The Proposed Framework

Our framework is designed based on IDLP-RDIC, including lightweight stream data integrity checking and secure data updates for IoT.

A. Architecture

The architecture of our framework is depicted in Fig. 1, which can be generally divided into four parts, including the IoT, the verifiers, the PMT, and the RS:

1) *The IoT.* The IoT is the origin of the stream files. The nodes split the files into blocks and generate HVTs, then outsource them to the RS. When the nodes intend to check the integrity or update their files, they generate requests and forwards them to the verifiers.
2) *The Verifiers.* The multiple verifiers are located on the edge of the IoT. They receive the checking and updating requests from the IoT nodes, generate challenges, and send them to the RS. They also receive the proofs of integrity from the RS, check their validity, record these interactions into the IDBCs, then send the results to the nodes and RS.
3) *The PMN.* The management organization (MO) which operates the entire architecture deploys the PMN, which consists of the mobile base stations and Wi-Fi hotspots, provided by the devices like UAVs and vehicles. It plays the role of the PKG in IDLP-RDIC, i.e., it generates a master private key and a corresponding master public key for the entire architecture, receives the identities (IDs) from the entities (except the RS) during the registrations, then sends the corresponding private keys along with the master public key. This is a one-time step for each entity.
4) *The RS.* The RS is located on the edge of the IoT or on the Internet. It stores the nodes' stream files and HVTs. After receiving the challenges from the verifiers, it performs the updates (if there are), generates the proofs of integrity by aggregating the challenged blocks and the corresponding HVTs according to IDLP-RDIC.

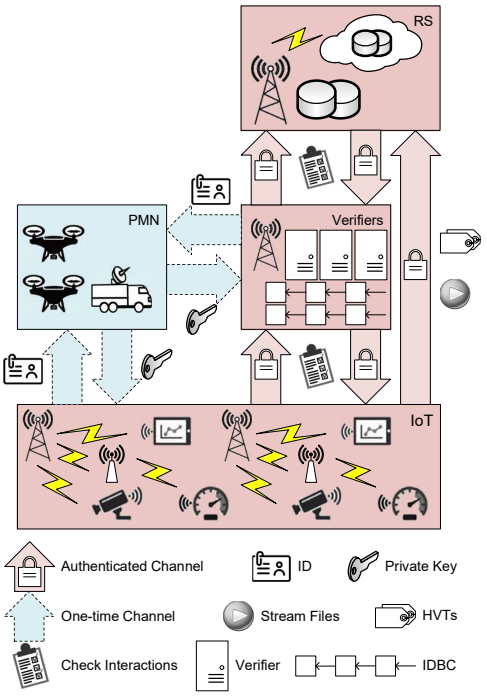

Fig. 1. Architecture of the framework.

As depicted in Fig. 1, the only trusted entity is the PMN with the one-time channels used for delivering keys. All the other entities are assumed to be public and may be suffered from malicious insiders and outsiders. However, the messages transferred via the public channels are still authenticated by digital signatures.

We firstly define the notations as listed in Table I, which will be used in the rest of this paper. Then, we make the following assumptions:

1) N_i and V_d, V_1, V_2, ... extract their private keys from the PMN by calling $Extract()$ and the RS publishes its public key, which is the only one public key used in the framework except mpk. We omit these one-time steps for simplicity.
2) All the messages are cryptographically authenticated, i.e., if N_i sends any message Msg_i to another entity, it should also send the corresponding $IDS_i(Msg_i)$, so as the entity can authenticate Msg_i by verifying $IDS_i(Msg_i)$ using NID_i. V_d, V_1, V_2, ... work in the same way, whereas for the RS which is not reachable by the PMN, if it sends any message Msg_{RS} to another entity, it should also send the corresponding $Sig_{RS}(Msg_{RS})$, so as the entity can authenticate

TABLE I
NOTATIONS

Notation	Description
N_i	The i-th node
V_d	The designated verifier
$V_1, V_2, \ldots$	The witness verifiers
PMN	The private mobile network
RS	The remote storage
MO	The management organization
$IDBC_i$	The identity-based blockchain for the i-th node
NID_i	The ID of the i-th node
FID_{ij}	The name of the j-th file of the i-th node
$OP_{i,\hat{j}}$	The $\hat{j}$-th operation of N_i, possible values: $\{(FID, \text{VERIFY}), (FID, \text{CREATE}, n), (FID, \text{APPEND}, n), (FID, \text{DELETE})\}$
ts, ts'	The time-stamps
R	The request from N_i, which equals to $(NID_i, \{OP_{ij}\}, ts)$
IDS	The identity-based signature
Sig	The traditional digital signature
τ	The temporary key
$\{m'_{ijk}\}$	The new file blocks
$\{t'_{ijk}\}$	The HVTs of the new file blocks
E_i	The extra authentication information
$\{m_{ijk}\}$	The hosted file blocks
$\{t_{ijk}\}$	The HVTs of the hosted file blocks
n_{ij}	The length of FID_{ij} (in blocks)
P'	The proof of integrity (self-check)
P	The proof of integrity
PoV	The proof of verification
$Result$	The result of the operations, possible values: {NULL,UNREASONABLE,VALID,INVALID}

Msg_{RS} by verifying $Sig_{RS}(Msg_{RS})$ using the RS's public key.

3) ts or ts' uniquely identifies a request or an acknowledgment to defend replay attacks. The offset between the time-stamp and the synchronized global time must be smaller than a specific value.

As depicted in Fig. 2, our framework consists of seven algorithms, which are described by Algorithm 1 to 7 as follows:

1) *REQUEST_AND_UPLOAD.* N_i generates a list of operations, selects V_d, generates the request R, and forwards R to V_d via the MO. N_i also generates the new file blocks and forwards them to the RS along with the HVTs.
   Remarks:
   a) Batch operations are supported, i.e., $\{OP_{i,\hat{j}}\}$ in R may be multiple. However, to confirm the consistency, simultaneous requests for a same node are prevented by the MO.
   b) $\{FID_{ij}\}$ and $\{n_{ij}\}$ are not explicitly recorded, but can be derived from $\{i, \hat{j}\}$ in $IDBC_i$. For instance, after the sequence $(FID_{1,3}, \text{CREATE}, 10)$, $(FID_{1,3}, \text{APPEND}, 10)$, and $(FID_{1,1}, \text{DELETE})$, $\{FID_{1,1}, FID_{1,2}\}$ becomes $\{FID_{1,2}, FID_{1,3}\}$, and $\{n_{1,1} = 10, n_{1,2} = 10\}$ becomes $\{n_{1,2} = 10, n_{1,3} = 20\}$, respectively. The initial values of $\{FID_{ij}\}$ and $\{n_{ij}\}$ are both empty.
   c) The criteria for selecting V_d may be arbitrary, such as the latency, location, computing power, or even reputation.
2) *CHALLENGE.* On receiving R, V_d sends an acknowledgment to N_i. If $IDS_i(R)$ and ts are valid but any $OP_{i,\hat{j}}$ is invalid, V_d declares that N_i is dishonest, then records the unreasonable request in $IDBC_i$ (by jumping to Algorithm 5). Otherwise, V_d sends $(R, IDS_i(R))$ to the RS along with a random τ as a challenge token.
   Remarks:
   a) When $OP_{i,\hat{j}} == (FID_{ij}, \text{CREATE}, n)$, $OP_{i,\hat{j}}$ is reasonable if and only if: FID_{ij} has not been created, no matter whether it has been deleted, in $IDBC_i$ or $\{OP_{i,1}, \ldots, OP_{i,\hat{j}-1}\}$. (We note that the deleted FIDs are not reusable. Otherwise, a security issue may occur [15].)
   b) When $OP_{i,\hat{j}} \in \{(FID_{ij}, \text{VERIFY}), (FID_{ij}, \text{APPEND}, n), (FID_{ij}, \text{DELETE})\}$, $OP_{i,\hat{j}}$ is rea-

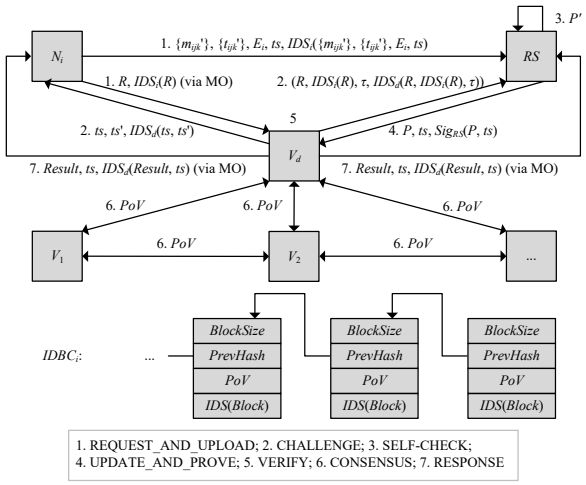

Fig. 2. Interactions of the algorithms.

Algorithm 1: REQUEST_AND_UPLOAD

Run by: N_i
Retrieve $\{FID_{ij}\}$ and $\{n_{ij}\}$ from $IDBC_i$;
foreach $\hat{j}$ **do**
  Set $OP_{i,\hat{j}}$ as one of $(FID_{ij}, \text{VERIFY})$, $(FID_{ij}, \text{CREATE}, n)$, $(FID_{ij}, \text{APPEND}, n)$, $(FID_{ij}, \text{DELETE})$;
  if $OP_{i,\hat{j}} == (FID_{ij}, \textit{CREATE}, n)$ ***or*** $OP_{i,\hat{j}} == (FID_{ij}, \textit{APPEND}, n)$ **then**
    Split the new data (to be created as or appended to FID_{ij}) to $\{m'_{ijk}\}$;
    Call $HVT - Gen(NID_i, FID_{ij}, \{m'_{ijk}\}, s, sk_i)$ to get $\{t'_{ijk}\}$ and E_i;
  end
  Select V_d;
  Generate ts;
  Set $R = (NID_i, \{OP_{ij}\}, ts)$;
  Send R and $IDS_i(R)$ to MO, MO records and forwards them to V_d if all the previous requests from N_i are finished;
  Send $(\{m'_{ijk}\}, \{t'_{ijk}\}, E_i, ts)$ and $IDS_i(\{m'_{ijk}\}, \{t'_{ijk}\}, E_i, ts)$ to RS;
end

Algorithm 2: CHALLENGE

Run by: V_d
if $IDS_i(R)$ *is valid* ***and*** ts *is valid* **then**
  Generate ts';
  Send the acknowledgment $(ts, ts', IDS_d(ts, ts'))$ to N_i;
  Retrieve $\{FID_{ij}\}$ and $\{n_{ij}\}$ from $IDBC_i$;
  foreach $\hat{j}$ **do**
    if $OP_{i,\hat{j}}$ *is unreasonable* **then**
      Set *Result*=UNREASONABLE;
      goto Algorithm 5;
    end
  end
  Select a random $\tau \in Z_q$;
  Send $(R, IDS_i(R), \tau)$ and $IDS_d(R, IDS_i(R), \tau)$ to RS;
else
  Drop R and $IDS_i(R)$, then ask for a re-transmission;
end

sonable if and only if: FID_{ij} has been created, but has not been deleted in $IDBC_i$ and $\{OP_{i,1}, \ldots, OP_{i,\hat{j}-1}\}$.

3) *SELF-CHECK.* On receiving the challenge token from V_d and the new file blocks along with the HVTs from N_i, the RS checks their validity. If all the signatures and time-stamps are valid but any $OP_{i,\hat{j}}$ is invalid, the RS declares that N_i and V_d are both dishonest and conspiring with each other, then reports them to the MO. If $(\{m'_{ijk}\}, \{t'_{ijk}\})$ does not matches I' (i.e., any index $(i, j) \notin I'$), or the self-generated proof of integrity P' is invalid, the RS declares that N_i is dishonest, then reports N_i to the MO.
4) *UPDATE_AND_PROVE.* If the challenge token, the new file blocks, and the HVTs are all valid, the RS performs the operations. Then, it generates the proof of integrity P, from the hosted file blocks indicated by $\{(FID_{ij}$, VERIFY$)\}$ and the new file blocks $\{m'_{ijk}\}$ altogether. No matter how many file blocks are referred, only one proof is generated. Then, the RS forwards P to V_d.
5) *VERIFY.* V_d generates PoV. If $Result ==$ UNREASONABLE (from Algorithm 2), V_d sets $PoV = (R, Result, IDS_i(R), IDS_d(Result, ts))$, otherwise V_d sets $PoV = (R, \tau, P, Result, IDS_i(R), IDS_d(R, IDS_i(R), \tau), Sig_{RS}(P, ts), IDS_d(Result, ts))$ (from Algorithm 4). Then, V_d generates a new block for $IDBC_i$ and broadcasts it to V_1, V_2, ...
6) *CONSENSUS.* V_1, V_2, ... check the new block received from V_d. If $BlockSize$, $PrevHash$, PoV, and $IDS_d(Block)$ are all valid, they append the new block to their local $IDBC_i$, then send confirmations to V_d.
7) *RESPONSE.* If the new block is confirmed by more than half of the verifiers, V_d sends the result to N_i and the RS via the MO.

Algorithm 3: SELF-CHECK

Run by: RS
if $IDS_d(R, IDS_i(R), \tau)$ *is valid* ***and*** ts *is valid* **then**
  if $IDS(\{m'_{ijk}\}, \{t'_{ijk}\}, E_i, ts)$ *is valid* ***and*** ts *matches* R **then**
    if *all* $OP_{i,\hat{j}}$ *are reasonable* **then**
      Set $I' = \{\}$;
      foreach $\hat{j}$ **do**
        if $OP_{i,\hat{j}} \in \{(FID_{ij}$, *CREATE*, $n)$, $(FID_{ij}$, *APPEND*, $n)\}$ **then**
          for $k = 1$ **to** n_{ij} **do**
            Set $I' = I' \cup (j, k)$;
          end
        end
      end
      if $(\{m'_{ijk}\}, \{t'_{ijk}\})$ *matches* I' **then**
        Retrieve the hosted $\{FID_{ij}\}$ and $\{n_{ij}\}$;
        Select a random $\tau' \in Z_q$;
        Set $C' = (I', \tau')$;
        Call $ProofGen(C', \{m'_{ijk}\}, \{t'_{ijk}\}, E_i, s)$ to get P';
        if $ProofCheck(NID_i, \{FID_{ij}\}, C', P', s, mpk)$==*INVALID* **then**
          Report N_i to MO by submitting R, $IDS_i(R)$, $(\{m'_{ijk}\}, \{t'_{ijk}\}, E_i, ts)$, and $IDS_i(\{m'_{ijk}\}, \{t'_{ijk}\}, E_i, ts)$;
        else
          goto Algorithm 4;
        end
      else
        Report N_i to MO by submitting R, $IDS_i(R)$, $(\{m'_{ijk}\}, \{t'_{ijk}\}, E_i, ts)$, and $IDS(\{m'_{ijk}\}, \{t'_{ijk}\}, E_i, ts)$;
      end
    else
      Report both N_i and V_d to MO by submitting $(R, IDS_i(R), \tau)$ and $IDS_d(R, IDS_i(R), \tau)$;
    end
  else
    Drop $(\{m'_{ijk}\}, \{t'_{ijk}\}, E_i, ts)$ and $IDS_i(\{m'_{ijk}\}, \{t'_{ijk}\}, E_i, ts)$, then ask for a re-transmission;
  end
else
  Drop $(R, IDS_i(R), \tau)$ and $IDS_d(R, IDS_i(R), \tau)$, then ask for a re-transmission;
end

V. Security Analysis

A. Threat Model

In our framework, the MO and PMN (with the one-time channels) are trusted. For the other entities, we consider threats from four different aspects as follows.

1) The untrusted RS may conceal data errors by forging proofs of integrity to deceive the verifiers. In addition, it may not perform the updates correctly.
2) The untrusted verifiers may be unavailable (due to connections lost or even crashed down) or dishonest (if invalid challenge tokens is submitted to the RS or forge checking results to deceive the nodes). In addition, they are curious about the nodes' data.
3) The malicious nodes may deliberately submit invalid HVTs and perform unreasonable operations to the RS, then accuse the RS of damaging their data.
4) The external adversaries may defer and tamper with the communications between the entities.

For the adversarial capabilities, we make the following assumptions:

1) The RS is always available, i.e., it always responds to the other entities within a reasonable period of time. In addition, the RS is trusted in data privacy, i.e., it has no

Algorithm 4: UPDATE_AND_PROVE

Run by: RS
Sets $I = \{\}$;
foreach $\hat{j}$ **do**
  if $OP_{i,\hat{j}} == (FID_{ij}, \textit{VERIFY})$ **then**
    for $k = 1$ **to** n_{ij} **do**
      Set $I = I \cup (j, k)$;
    end
  end
  if $OP_{i,\hat{j}} == (FID_{ij}, \textit{CREATE}, n)$ **then**
    Stores FID_{ij} and n_{ij};
    for $k = 1$ **to** n_{ij} **do**
      Store m'_{ijk} as m_{ijk};
      Store t'_{ijk} as t_{ijk};
      Set $I = I \cup (j, k)$;
    end
  end
  if $OP_{i,\hat{j}} == (FID_{ij}, \textit{APPEND}, n)$ **then**
    Set $n_{ij} = n_{ij} + n$; **for** $k = n_{ij} + 1$ **to** $n_{ij} + n$ **do**
      Store m'_{ijk} as m_{ijk};
      Store t'_{ijk} as t_{ijk};
      Set $I = I \cup (j, k)$;
    end
  end
  if $OP_{i,\hat{j}} == (FID_{ij}, \textit{DELETE})$ **then**
    for $k = 1$ **to** n_{ij} **do**
      Delete the hosted $\{m_{ijk}\}$ and $\{t_{ijk}\}$;
    end
    Delete the hosted n_{ij} and FID_{ij};
  end
end
Set $C = (I, \tau)$;
Call $ProofGen(C, \{m_{ijk}\}, \{t_{ijk}\}, E_i, s)$ to get P;
Send (P, ts) and $Sig_{RS}(P, ts)$ to V_d;

Algorithm 5: VERIFY

Run by: V_d
if *Result==UNREASONABLE* **or** *($Sig_{CS}(P, ts)$ is valid* **and** *ts is valid)* **then**
  if *Result==UNREASONABLE* **then**
    Generate $IDS_d(Result, ts)$;
    Set $PoV = (R, Result, IDS_i(R), IDS_d(Result, ts))$;
  else
    Set $I = \{\}$;
    foreach $\hat{j}$ **do**
      if $OP_{i,\hat{j}} == (FID_{ij}, \textit{VERIFY})$ **then**
        for $k = 1$ **to** n_{ij} **do**
          Set $I = I \cup (j, k)$;
        end
      end
      if $OP_{i,\hat{j}} == (FID_{ij}, \textit{CREATE}, n)$ **then**
        Add FID_{ij} to $\{FID_{ij}\}$;
        for $k = 1$ **to** n_{ij} **do**
          Set $I = I \cup (j, k)$;
        end
      end
      if $OP_{i,\hat{j}} == (FID_{ij}, \textit{APPEND}, n)$ **then**
        for $k = n_{ij} + 1$ **to** $n_{ij} + n$ **do**
          Set $I = I \cup (j, k)$;
        end
      end
    end
    Set $C = (I, \tau)$;
    Set $Result = ProofCheck(NID_i, \{FID_{ij}\}, C, P, s, mpk) \in \{\text{VALID, INVALID}\}$;
    Generate $IDS_d(Result, ts)$;
    Set $PoV = (R, \tau, P, Result, IDS_i(R), IDS_d(R, IDS_i(R), \tau), Sig_{RS}(P, ts), IDS_d(Result, ts))$;
  end
  Generate a new block $(BlockSize, PrevHash, PoV, IDS_d(Block))$;
  Append the new block to V_d's local blockchain;
  Broadcast the new block to V_1, V_2, ...;
else
  Drop (P, ts) and $Sig_{RS}(P, ts)$, then ask for a re-transmission;
end

incentive to reveal the data because of the commercial interests.

2) More than half the verifiers are available and honest. The designated verifier who has performed its duty honestly will receive rewards from the MO.
3) The external adversaries cannot delay the communications indefinitely. The re-transmissions of timeout can ensure that all the messages reach their destinations within a reasonable period of time.

B. Security Properties

Under the threat model presented above, our framework is reliable, according to the following security properties.

1) In IDLP-RDIC, with ensured by the hardness of the CDH problem and DL problem, the RS who convinced the verifiers actually stores all the challenged blocks, i.e., there is a polynomial-time algorithm to retrieve all the challenged blocks from the RS. Meanwhile, the proof of integrity is masked by a random element during each verification. Hence, the verifiers cannot derive any content of the nodes' data. These properties are summarized as completeness, soundness, and privacy-preserving, which will be rigorously proved in Appendix A.
2) With ensured by the soundness of IDLP-RDIC, if the RS has damaged, deleted, or just incorrectly updated the data, it cannot generate a valid proof of integrity (Algorithm 4). Any invalid proof will be detected by the verifiers and recorded in the IDBCs as bad credit (Algorithm 5).
3) To win the rewards, a reasonable verifier who has accepted the task from a node will perform his duty honestly. Even if the designated verifier submits an invalid challenge token to the RS or lies about the result, the task could not be recorded in the IDBCs (Algorithm 6). After a period of time, if the node cannot query the result, it will report the designated verifier to the MO by submitting the acknowledgment along with the signature.
4) Any unreasonable operation (such as creating a file using an existing FID, appending to or deleting a file using an uncreated or deleted FID) submitted by a malicious node will be detected by the designated verifier (Algorithm 2) and recorded in the IDBCs as bad credit (Algorithm 5). Any invalid HVT uploaded by a malicious node will be detected by the RS, who will report the node to the MO, by submitting the invalid HVTs along with the signature (Algorithm 3). Even if the node conspires with the designated verifier, the unreasonable operations and invalid challenge tokens

Algorithm 6: CONSENSUS

Run by: $V_1, V_2, \ldots$
Check $BlockSize$, $PrevHash$, and $IDS_d(Block)$;
Check $IDS_i(R)$, $IDS_d(Result, ts)$, and ts;
if $Result \in \{VALID, INVALID\}$ **then**
  Check $IDS_d(R, IDS_i(R), \tau)$;
  Check $Sig_{RS}(P, ts)$;
  Retrieve $\{FID_{ij}\}$ and $\{n_{ij}\}$ from $IDBC_i$;
  Set $I = \{\}$;
  foreach $\hat{j}$ **do**
    if $OP_{i,\hat{j}} == (FID_{ij}, VERIFY)$ **then**
      for $k = 1$ **to** n_{ij} **do**
        Set $I = I \cup (j, k)$;
      end
    end
    if $OP_{i,\hat{j}} == (FID_{ij}, CREATE, n)$ **then**
      Add FID_{ij} to $\{FID_{ij}\}$;
      for $k = 1$ **to** n_{ij} **do**
        Set $I = I \cup (j, k)$;
      end
    end
    if $OP_{i,\hat{j}} == (FID_{ij}, APPEND, n)$ **then**
      for $k = n_{ij} + 1$ **to** $n_{ij} + n$ **do**
        Set $I = I \cup (j, k)$;
      end
    end
  end
  Set $C = (I, \tau)$;
  Check if $Result == ProofCheck(NID_i, \{FID_{ij}\}, C, P, s, mpk)$;
end
if *all valid* **then**
  Append the new block to their local blockchain;
  Response the confirmation to V_d;
else
  Drop the new block and ask for a re-transmission;
end

Algorithm 7: RESPONSE

Run by: V_d
Wait for the responses from V_1, V_2, ..., re-transmit PoV if necessary;
if *the new block is confirmed by more than half the verifiers* **then**
  Send $(Result, ts)$ and $IDS_d(Result, ts)$ to MO, MO records and forwards them to N_i and RS;
end

will be also detected by the RS, who will report both the node and designated verifier to the MO, by submitting both the invalid request and challenge token along with the signatures (Algorithm 3).

5) The public channels are all cryptographically authenticated. Any message without a valid signature or timestamp will be dropped and re-transmitted, so that the external adversaries cannot tamper with the communications.

VI. Evaluation

A. Theoretical Evaluation

1) Computation Costs: We compare the computation costs of IDLP-RDIC with two dynamic RDIC schemes (Wang *et al.*'s scheme [20] and MuR-DPA [21]) and three identity-based RDIC schemes (ID-DPDP [12], ID-RDIC [13], and Shen *et al.*'s scheme [15]). To be fair, we do not consider any extra cost invoked by a special feature (e.g., multi-replica verification in MuR-DPA or sensitive data hiding in Shen's scheme). We summarize the comparisons in Table II. $|F|$ and $|F'|$ denote the sizes of the entire file and the challenged part of the entire file, respectively; S denotes the sector size; s denotes the sector number; Exp and Exp_T denote the time costs of single exponentiation on G and G_T, respectively; $Hash$ and $Hash_T$ denote the time costs of single hash to G and G_T, respectively; $Pair$ denotes the time cost of single bilinear pairing, Sig denotes the time cost of generating or verifying a signature. Comparing to Exp, Exp_T, $Hash$, $Hash_T$, $Pair$, and Sig, the other operations such as multiplications on G, G_T and Z_q, additions on Z_q, and hashes to Z_q are negligible. To show why so is, we run these operations separately for 1000 times on a Raspberry Pi 4 and an NVIDIA Jetson Nano, then summarize the time costs in Table III.

It is evident that the HVT-generation of IDLP-RDIC is the fastest because it requires only $\frac{2F}{Ss}$ exponentiation operations on G, i.e., for the same file size, the larger the sector number and the sector size, the faster the speed. For the proof-generation and proof-check phases, IDLP-RDIC is slightly slower than those of ID-DPDP and MuR-DPA, but the differences are constant.

2) Communication Costs: We compare the communication costs of IDLP-RDIC with the schemes mentioned above. We do not consider any extra cost invoked by a special feature either. The communication costs of these schemes are mainly from the proof-generation phases, as shown in Table IV. $|G|$, $|Z_q|$, and $|G_T|$ denote the sizes of an element in G, Z_q, and G_T, respectively; $|Sig|$ denotes the size of a signature; s denotes the sector number.

We can see that all of these schemes achieve the fundamental property of RDIC, i.e., the proof of integrity is independent to the size of the challenged data. Although larger than the other schemes, the communication cost of IDLP-RDIC is still constant for a fixed s, regardless of the data size.

3) Blockchain Properties: We compare the blockchain properties of our framework with four related blockchain-based systems from seven aspects, (a) blockchain type: public, consortium, or private; (b) maximum tolerance: the maximum proportion of faulty nodes which the blockchain tolerates; (c) Byzantine fault-tolerance: whether the blockchain tolerates Byzantine faults; (d) privacy-preserving, whether the blockchain leaks the sensitive personal data; (e) no mining: whether the consensus mechanism does not require mining operations; (f) communication complexity: the bandwidth cost for achieving consensus; (g) no PKI: whether the consensus mechanism does not require PKI. We summarize the comparisons in Table V.

We can see that our framework realizes Byzantine fault tolerance without mining or PKI, because all the information embedded in a block is simultaneously signed by the nodes, the designated verifier, and the RS, along with a unique timestamp. For these signatures, the blocks can be generated and verified publicly and efficiently, without any extra cost invoked by a traditional consensus mechanism, such as PoW or PBFT.

TABLE II
COMPARISON OF COMPUTATION COSTS

Scheme	HVT-generation	Proof-generation	Proof-check
Wang *et al.* [20]	$\frac{2\|F\|}{S}Exp + \frac{\|F\|}{S}Hash$	$\frac{\|F'\|}{S}Exp$	$(\frac{\|F'\|}{S}+1)Exp + \frac{\|F'\|}{S}Hash + 2Pair$
IP-DPDP [12]	$(\frac{\|F\|}{S} + \frac{\|F\|}{Ss})Exp + \frac{\|F\|}{Ss}Hash$	$\boldsymbol{\frac{\|F'\|}{Ss}Exp}$	$\boldsymbol{(\frac{\|F'\|}{Ss} + s)Exp + \frac{\|F'\|}{Ss}Hash + 2Pair}$
MuR-DPA [21]	$(\frac{\|F\|}{S} + \frac{\|F\|}{Ss})Exp + \frac{\|F\|}{Ss}Hash$	$\boldsymbol{\frac{\|F'\|}{Ss}Exp}$	$\boldsymbol{(\frac{\|F'\|}{Ss} + s)Exp + \frac{\|F'\|}{Ss}Hash + 2Pair}$
ID-RDIC [13]	$(\frac{\|F\|}{S}+1)Exp + \frac{\|F\|}{S}Hash + 1Sig$	$\frac{\|F'\|}{S}Exp + 1Exp_T + 1Hash_T$	$(\frac{\|F'\|}{S}+1)Exp + \frac{\|F'\|}{S}Hash + 1Pair + 1Sig$
Shen *et al.* [15]	$\frac{2\|F\|}{S}Exp + \frac{\|F\|}{S}Hash + 1Sig$	$\frac{\|F'\|}{S}Exp$	$(\frac{\|F'\|}{S}+1)Exp + \frac{\|F'\|}{S}Hash + 4Pair + 1Sig$
IDLP-RDIC	$\boldsymbol{\frac{2\|F\|}{Ss}Exp + \frac{\|F\|}{Ss}Hash}$	$\frac{\|F'\|}{Ss}Exp + 1Pair + 1Exp_T$	$(\frac{\|F'\|}{Ss}+s+1)Exp + \frac{\|F'\|}{Ss}Hash + 2Pair + 1Exp_T + 1Sig$

TABLE III
COMPARISON OF TIME COSTS OF DIFFERENT OPERATIONS

Operation (1000 Times)	Time cost (ms)	
	Raspberry Pi 4	NVIDIA Jetson Nano
Hash to G ($Hash$)	11300.56	8919.44
Pairing ($Pair$)	6270.13	3531.85
Exponentiation on G (Exp)	4885.11	3952.40
Hash to G_T ($Hash_T$)	1020.94	568.03
Exponentiation on G_T (Exp_T)	710.57	425.43
Multiplication on G	21.36	17.77
Multiplication on G_T	4.63	2.75
Hash to Z_q	1.13	0.96
Multiplication on Z_q	0.22	0.15
Addition on Z_q	0.04	0.04

TABLE IV
COMPARISON OF COMMUNICATION COSTS OF PROOF-GENERATION

Scheme	Cost
Wang *et al.* [20]	$1\|G\| + 1\|Z_q\|$
IP-DPDP [12]	$1\|G\| + s\|Z_q\|$
MuR-DPA [21]	$1\|G\| + s\|Z_q\|$
ID-RDIC [13]	$1\|G\| + 1\|G_T\| + 1\|Sig\|$
Shen *et al.* [15]	$1\|G\| + 1\|Z_q\|$
IDLP-RDIC	$(s+3)\|G\| + s\|Z_q\| + 1\|G_T\|$

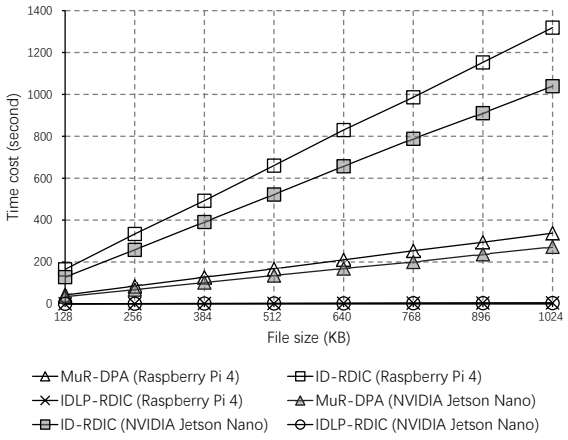

Fig. 3. Comparisons for time costs for HVT-generation.

B. Experimental Performance

1) Environment: For the hardware environment, we simulate two IoT nodes using a Raspberry Pi 4 and an NVIDIA Jetson Nano, respectively. Raspberry Pi is a very cheap SBC (single board computer), which provides a set of GPIO (general purpose input/output) pins to control electronic components for IoT. NVIDIA Jetson Nano is also an SBC, which is designed for small, power-efficient AI systems and embedded IoT applications. Both of them are lightweight devices equipped with ARM CPUs. We simulate one PMN, one RS, and three verifiers on five virtual machines (VM) separately, provided by VMware Workstation which is running on a server equipped with an Intel Xeon Silver 4114 CPU and 32 GB DDR4 memory.

For the network parameters, according to some recent reports on modern 5G networks [34], [35], we set the peak incoming/outgoing data rate of each VM to 1 Gb/s. We also set the incoming/outgoing latency for the verifiers to 15 ms, while 30 ms for the RS, for it is usually located remotely. Then, we connect the Raspberry Pi and NVIDIA Jetson Nano to the simulated network via an 802.11ac Wi-Fi router. The measured performance is summarized in Table VI.

For the software environment, we install Raspbian Buster, Ubuntu 18.04, Windows 10, and Debian 10 on the Raspberry Pi 4, the NVIDIA Jetson Nano, the server, and the VMs, respectively. We use a Type A elliptic curve with 160-bit group order provided by PBC Library [36] as the bilinear pairing, use SHA-256 provided by OpenSSL [37] as the cryptographic hash function, use Cha-Cheon's scheme [38] as the IDS, and use BLS signature [24] as the traditional digital signature, respectively.

2) Results: We firstly compare the time costs of IDLP-RDIC with MuR-DPA and ID-RDIC for the most computational-expensive HVT-generation phases, on the Raspberry Pi and NVIDIA Jetson Nano respectively, in Fig. 3. Because both MuR-DPA and ID-RDIC are not suitable for lightweight devices, we select a relatively smaller dataset (various from 128 KB to 1 MB), otherwise they will be extremely slow. (For a 1 GB file on a Raspberry Pi 4, it takes MuR-DPA and ID-RDIC about 100 hours and more than 300 hours, respectively.) Limited by the group order (160 bits), the smallest data unit must be less than 20 bytes. For MuR-DPA and IDLP-RDIC, we set the sector size to 16 bytes and the block size to 64 sectors (1024 bytes). For ID-RDIC, we set the block size to 16 bytes, because a block cannot be further divided. All the results represent the means of 10 trails.

In addition, we would like to provide the computation and

TABLE V
COMPARISON OF BLOCKCHAIN PROPERTIES

Framework	Blockchain Type	Maximum Tolerance	Byzantine Fault-tolerance	Privacy-preserving	No Mining	Communication Complexity	No PKI
Bitcoin [25]	Public	50%	Yes	No	No	$O(N)$[a]	Yes[b]
DeepCoin [31]	Consortium	33%	Yes	Yes	Yes	$O(N^2)$	No
Dwivedi *et al.* [32]	Consortium	50%	Yes	Yes	Yes	$O(N)$	No
Derhab *et al.* [33]	Private	50%	No[c]	No	Yes	$O(N)$	No
Ours	Consortium	50%	Yes	Yes	Yes	$O(N)$	Yes

[a] N denotes the number of participants.
[b] The public keys are embedded into the blocks.
[c] The participants are assumed to be trusted.

TABLE VI
NETWORK PARAMETERS

Route	Bandwidth (Mb/s)[a]	Latency (ms)[b]
Raspberry Pi to Verifier	636	33.8
Raspberry Pi to RS	367	65.8
NVIDIA Jetson Nano to Verifier	662	32.5
NVIDIA Jetson Nano to RS	373	65.6
Verifier to Verifier	339	62.7
Verifier to RS	238	93.8

[a] Measured by iPerf (TCP mode).
[b] Measured by Ping.

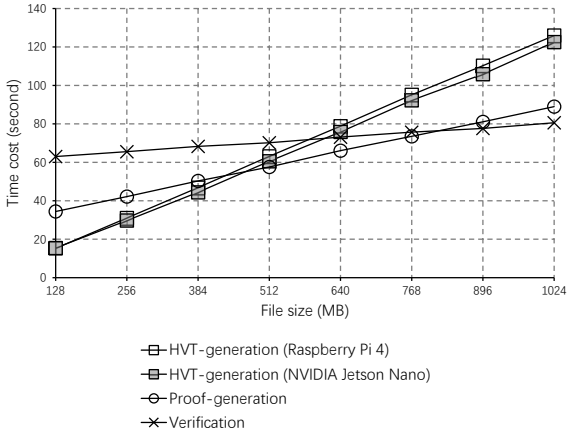

Fig. 4. Time costs of our framework for different phases.

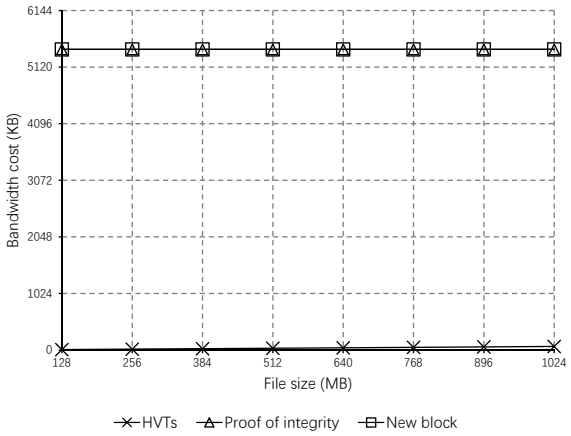

Fig. 5. Bandwidth costs of our framework for different parts.

communication costs of our framework. Fig. 4 illustrates the time costs of different phases. Here, the verification phase means, the designated verifier checks the proof of integrity and broadcasts the PoV to the other verifiers, the other verifiers check the PoV, and the IDBC network reaches consensus. Fig. 5 illustrates the bandwidth costs for transferring the HVTs, proofs of integrity, and new blocks (for the IDBCs) during each simulation. In these cases, the Raspberry Pi 4 or NVIDIA Jetson Nano generates HVTs for much larger files (various from 128 MB to 1 GB) and uploads them to the RS. Then, the verifiers check their integrity. We set the sector size to 16 bytes and the block size to 65,536 sectors (1 MB). For each block size, we repeat the simulation for 10 times and all the results represent the mean values. We do not analyze the costs of updates here, because they just equal to the costs of creating the files which are comprised of the updated blocks.

3) Discussion: We would like to discuss the aforementioned results. Fig. 3 demonstrates that the HVT-generation of IDLP-RDIC is much faster than those of MuR-DPA and ID-RDIC. For a certain block size, the time cost of our HVT-generation is nearly independent of the sector number, i.e., it can be significantly decreased by increasing the sector number of each block, while MuR-DPA and ID-RDIC do not benefit from this method.

Fig. 4 further demonstrates the high computational efficiency of our framework. For a 1 GB file, the HVT-generation takes about 126 seconds (8.1 MB/s) on the Raspberry Pi 4 and 123 seconds (8.3 MB/s) on the NVIDIA Jetson Nano, respectively, where the proof-generation takes about 89 seconds (11.5 MB/s) on a VM running on an Intel Xeon Silver 4114. It means that even for a 1080p video encoded in BluRay H.264 (about 4.58 MB/s), our framework can still work on-the-fly. After the file is uploaded, the subsequent verification task (from the designated verifier checking the proof of integrity to the IDBC network reaching consensus) takes about 81 seconds, which is also acceptable for a non-real-time procedure. Moreover, our solution does not rely on the complicated certificate managements. We also remark that the time cost of verification increases slowly along with the file size.

Fig. 5 shows the communication performance of our frame-

work. For a 1 GB file, the total HVTs generated by the IoT node (whether the Raspberry Pi 4 or NVIDIA Jetson Nano) are only 65 KB; both the proof of integrity generated by the RS and the new block generated by the designated verifier are about 5.3 MB. In fact, for a certain sector number, the size of the proof is constant, and the size of the new block is nearly constant, which is affordable for a consortium blockchain network with fast channels. (Although the set $\{OP_{i,\hat{j}}\}$ is variable, it only makes up a tiny proportion.)

In summary, the experimental results indicate that our framework is suitable for an actual IoT environment.

VII. Conclusion and Future Work

This paper proposes a secure lightweight stream data outsourcing framework for IoT. It eliminates the complicated or even untrusted PKI, by introducing a private mobile network to distribute the private keys via the one-time channels. It is reliable under untrusted channels, nodes, RS, and even verifiers, with ensured by the identity-based signatures, multiple verifiers, and blockchains. It also protects the owners' privacy, by revealing no information of the data content to any other entities besides the RS. Our simulations implicates that the framework is efficient for lightweight devices using ARM CPUs, thus it is suitable for an actual IoT.

The blockchain applied in our framework is not fully decentralized, for there must be a trusted MO to deploy the PMN. Building a secure and fully decentralized blockchain based data outsourcing framework may be a challenging issue, as well as an interesting future direction.

Appendix A
Security Proofs of IDLP-RDIC

In this appendix, we prove the security properties of IDLP-RDIC, including completeness, soundness, and privacy-preserving.

A. Completeness

The completeness of IDLP-RDIC is guaranteed by Theorem 1.

Theorem 1 (Completeness): IDLP-RDIC satisfies the following properties:

1) When the PKG sends a correct sk to the data owner, sk can pass the verification of the data owner.
2) When the RS properly stores the data user owner's file, the proof P it generates can pass the verification of the verifier.

Proof:

1) Given a private key $sk = (\alpha, \beta, \sigma)$, we have that:

$$\begin{aligned} & e(\sigma, g) \\ = \ & e(H_1(NID \parallel v \parallel u_1 \parallel \ldots \parallel u_s)^x, g) \\ = \ & e(H_1(NID \parallel g^{\alpha} \parallel g^{h_1(\beta)} \parallel \ldots \parallel g^{h_1(\beta+s-1)}), \\ & mpk). \end{aligned}$$

Then, the equation (1) holds. Therefore, sk passes the verification.

2) Given a correct proof $P = (M, T, E_i, \Gamma)$, firstly, we have that:

$$\begin{aligned} e(\sigma, g) &= e(H_1(NID \parallel v \parallel u_1 \parallel \ldots \parallel u_s)^x, g) \\ &= e(H_1(NID \parallel v \parallel u_1 \parallel \ldots \parallel u_s), mpk). \end{aligned}$$

Then, the equation (2) holds. Secondly, we have that:

$$\begin{aligned} & \Gamma \cdot e(T, g)^{\delta} \\ = \ & e(\textstyle\prod_{l=1}^{s} u_l, v)^{\gamma} \cdot e(\prod_{(j,k)\in I} {t_{jk}}^{a_{jk}}, g)^{\delta} \\ = \ & e(\textstyle\prod_{l=1}^{s} u_l, v)^{\gamma} \\ & \cdot e(\textstyle\prod_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}} \\ & \cdot \textstyle\prod_{(j,k)\in I} g^{\sum_{l=1}^{s} h_1(\beta+l-1) a_{jk} \tilde{m}_{jkl}}, v)^{\delta} \\ = \ & e(\textstyle\prod_{l=1}^{s} u_l, v)^{\gamma} \\ & \cdot e(\textstyle\prod_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}} \\ & \cdot \textstyle\prod_{l=1}^{s} {u_l}^{\sum_{(j,k)\in I} a_{jk} \tilde{m}_{jkl}}, v)^{\delta} \\ = \ & e(\textstyle\prod_{l=1}^{s} {u_l}^{\gamma}, v) \\ & \cdot e((\textstyle\prod_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\delta} \\ & \cdot \textstyle\prod_{l=1}^{s} {u_l}^{\delta \sum_{(j,k)\in I} a_{jk} \tilde{m}_{jkl}}, v) \\ = \ & e((\textstyle\prod_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\delta} \\ & \cdot \textstyle\prod_{l=1}^{s} {u_l}^{\gamma + \delta \sum_{(j,k)\in I} a_{jk} \tilde{m}_{jkl}}, v) \\ = \ & e((\textstyle\prod_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\delta} \\ & \cdot \textstyle\prod_{l=1}^{s} {u_l}^{\tilde{M}_l}, v). \end{aligned}$$

Then, the equation (3) holds. Therefore, P passes the verification.

This completes the proof of Theorem 1. ■

B. Soundness

Before the proof of the soundness, we first present the security model of IDLP-RDIC, i.e., the soundness game (Definition 1).

Definition 1 (Soundness Game): In IDLP-RDIC, for any probabilistic polynomial adversary $\mathcal{A}$ (malicious RS) and a challenger $\mathcal{C}$ (PKG, data owner, and verifier), the soundness game between $\mathcal{A}$ and $\mathcal{C}$ is described as follows:

1) $\mathcal{C}$ runs $Setup$ to obtain some public parameters $params$, a master private key x, a corresponding master public key mpk, a sector size S, and a sector number s, then forwards $params$, mpk, S, and s to $\mathcal{A}$, whereas keeps x secret.
2) $\mathcal{A}$ can make sort of $HVT - Gen$ queries to $\mathcal{C}$ for polynomial times adaptively, i.e., $\mathcal{A}$ can query the HVT t_{jk} for any NID, any file name FID_j, and any block m_{jk}. $\mathcal{C}$ computes the corresponding t_{jk} and forwards t_{jk} with the extra authentication information E to $\mathcal{A}$. We note that $\mathcal{A}$ is forbidden from making queries on the same NID, the same FID_j, and the same k with the

different $\{m_{jk}\}$. Furthermore, in $\mathcal{A}$'s view, each HVT is generated under a private key α using $u_1, \ldots, u_s$ (i.e., $t_{jk} = (H_2(NID \parallel FID_j \parallel k) \cdot \prod_{l=1}^{s} {u_l}^{\tilde{m}_{jkl}})^{\alpha}$) and can be verified using a public key $v = g^{\alpha}$, which v and $u_1, \ldots, u_s$ are all provided in E by the data owner rather than the PKG. ($\mathcal{A}$ never interacts with, or even cannot reach to PKG in IDLP-RDIC.) Therefore, we can safely forbid $\mathcal{A}$ from making any queries on the actual α and β.

3) For any NID, any set $\{FID_j\}$, and any set $\{m_{jk}\}$ on which $\mathcal{A}$ has made $HVT - Gen$ queries, $\mathcal{A}$ can undertake executions of $ProofGen$. In these executions, $\mathcal{C}$ plays the part of the verifier $\mathcal{V}$ and $\mathcal{A}$ plays the part of the prover $\mathcal{P}$. When an execution completes, $\mathcal{A}$ is provided with the output of $\mathcal{C}$. These executions can be arbitrarily interleaved with each other and with the queries mentioned above.
4) Eventually, $\mathcal{A}$ decides to be challenged on a certain NID^*, a certain set $\{FID_j\}$, and a certain set $\{m_{jk}\}$ with the indices $\{(j,k)\} = I^*$, that $\mathcal{A}$ has made $HVT - Gen$ queries on all of them. $\mathcal{A}$ outputs the description of a cheating prover $\mathcal{P}^*$. $\mathcal{P}^*$ is ϵ-admissible if it convincingly answers ϵ fraction of verification challenges, i.e., if $\Pr[\mathcal{V}(NID^*, \{FID_j\}, \{m_{jk}\}) \leftrightarrows \mathcal{P}^* = \text{VALID}] \geq \epsilon$ with $(j,k) \in I^*$. The probability here is over the coins of $\mathcal{A}$ and $\mathcal{C}$. $\mathcal{A}$ wins the game if he can successfully output an ϵ-admissible $\mathcal{P}^*$ when $1-\epsilon$ is negligible. We say IDLP-RDIC is sound if there exists an efficient extraction algorithm $Extr$ such that, for every adversary $\mathcal{A}$, whenever $\mathcal{A}$, playing the game defined above, outputs an ϵ-admissible $\mathcal{P}^*$ on NID^*, $\{FID_j\}$, and $\{m_{jk}\}$ with $(j,k) \in I^*$ when $1-\epsilon$ is negligible, $Extr$ recovers the blocks $\{m_{jk}\}$ from $\mathcal{P}^*$, i.e., $Extr(NID^*, \{FID_j\}, \mathcal{P}^*) = \{m_{jk}\}$ with $(j,k) \in I^*$, except possibly with negligible probability.

Then, we prove that IDLP-RDIC under this model in Theorem 2.

Theorem 2 (Soundness): If the CDH problem is hard in G, IDLP-RDIC is sound under Definition 1 in the random oracle model.

Proof: We prove this theorem as a series of games with interleaved analysis, inspired by [5].

Game 0. Game 0 is simply the soundness game defined in Definition 1.

Game 1. Game 1 is the same as Game 0, with one difference. $\mathcal{C}$ keeps a list of his responses to $HVT - Gen$ queries made by $\mathcal{A}$, and observes each response for $ProofGen$ executed by $\mathcal{A}$. If in any of these responses $\mathcal{A}$ is successful (i.e., $\mathcal{V}$ outputs VALID), but $\mathcal{A}$'s aggregated HVT $T' \neq \prod_{(j,k)\in I} {t_{jk}}^{a_{ik}}$, $\mathcal{C}$ declares failure and aborts.

Analysis. First, we can see that σ in E is actually a BLS signature [24] on $NID \parallel v \parallel u_1 \parallel \ldots \parallel u_s$ using the private key x and the equation (2) is the verification of σ using the public key mpk. According to the unforgeability of the BLS signature, we can safely assume that $u_1, \ldots, u_s$, and v used by $\mathcal{A}$ in the interactions and those in E have the same value, which is generated by $\mathcal{C}$ using NID, otherwise $\mathcal{V}$ outputs INVALID, except possibly with negligible probability.

Because the correct response satisfies the equation (3), we have:

$$\Gamma \cdot e(T, g)^{\delta}$$
$$= e((\prod\nolimits_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\delta} \cdot \prod\nolimits_{l=1}^{s} {u_l}^{\tilde{M}_l}, v).$$

If $T' \neq T$, we have:

$$\Gamma \cdot e(T', g)^{\delta}$$
$$= e((\prod\nolimits_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\delta} \cdot \prod\nolimits_{l=1}^{s} {u_l}^{\tilde{M}'_l}, v),$$

and at least one $\tilde{M}'_l \neq \tilde{M}_l$ (otherwise $T' = T$), therefore, if we define $\Delta\tilde{M}_l = \tilde{M}'_l - \tilde{M}_l$, at least one $\Delta\tilde{M}_l \neq 0$. We now show that if $\mathcal{A}$ causes $\mathcal{C}$ to abort with nonnegligible probability we can construct a simulator $\mathcal{S}$ that solves the CDH problem. $\mathcal{S}$ is given as inputs values $g, g^{\alpha}, h \in G$, its goal is to output h^{α}. $\mathcal{S}$ behaves like $\mathcal{C}$ in Game 0, with the following differences:

1) In the responses to the $HVT - Gen$ queries, $\mathcal{S}$ sets $v = g^{\alpha}$, randomly chooses $\{(\zeta_l, \eta_l) \mid l = 1, \ldots, s\}$, sets $\{u_l = g^{\zeta_l} h^{\eta_l} \mid l = 1, \ldots, s\}$, sets $\sigma = H_1(NID \parallel v \parallel u_1 \parallel \ldots \parallel u_s)^x$, randomly chooses $r_{jk}^{(NID)}$, programs the random oracle H_2 as $H_2(NID \parallel FID_j \parallel k) = g^{r_{jk}^{(NID)}} / (g^{\sum_{l=1}^{s} \zeta_l \tilde{m}_{jkl}} h^{\sum_{l=1}^{s} \eta_l \tilde{m}_{jkl}})$. Now $\mathcal{S}$ can compute:

$$\begin{aligned} t_{jk} &= (H_2(NID \parallel FID_j \parallel k) \cdot \prod\nolimits_{l=1}^{s} {u_l}^{\tilde{m}_{jkl}})^{\alpha} \\ &= (g^{r_{jk}^{(NID)}} / (g^{\sum_{l=1}^{s} \zeta_l \tilde{m}_{jkl}} h^{\sum_{l=1}^{s} \eta_l \tilde{m}_{jkl}}) \\ &\quad \cdot \prod\nolimits_{l=1}^{s} (g^{\zeta_l} h^{\eta_l})^{\tilde{m}_{jkl}})^{\alpha} \\ &= (g^{r_{jk}^{(NID)}} / (g^{\sum_{l=1}^{s} \zeta_l \tilde{m}_{jkl}} h^{\sum_{l=1}^{s} \eta_l \tilde{m}_{jkl}}) \\ &\quad \cdot (g^{\sum_{l=1}^{s} \zeta_l \tilde{m}_{jkl}} h^{\sum_{l=1}^{s} \eta_l \tilde{m}_{jkl}}))^{\alpha} \\ &= (g^{r_{jk}^{(NID)}})^{\alpha} \\ &= v^{r_{jk}^{(NID)}}. \end{aligned}$$

We note that guaranteed by the cryptographical hash function h_1, $\mathcal{A}$ cannot distinguish u_l from a random element without knowing β.

2) $\mathcal{S}$ continues interacting with $\mathcal{A}$ until the condition specified in Game 1 occurs. Dividing the verification equation for the forged T' and the expected T, we have:

$$\begin{aligned} e(T'/T, g)^{\delta} &= e(\prod\nolimits_{l=1}^{s} {u_l}^{\Delta\tilde{M}_l}, v) \\ &= e(\prod\nolimits_{l=1}^{s} (g^{\zeta_l} h^{\eta_l})^{\Delta\tilde{M}_l}, v) \\ &= e(\prod\nolimits_{l=1}^{s} (v^{\zeta_l} (h^{\alpha})^{\eta_l})^{\Delta\tilde{M}_l}, g) \\ &= e(v^{\sum_{l=1}^{s} \zeta_l \Delta\tilde{M}_l} (h^{\alpha})^{\sum_{l=1}^{s} \eta_l \Delta\tilde{M}_l}, g). \end{aligned}$$

We can see that we have found the solution the CDH problem $h^{\alpha} = (T'^{\delta} \cdot T^{-\delta} \cdot v^{-\sum_{l=1}^{s} \zeta_l \Delta\tilde{M}_l})^{(\sum_{l=1}^{s} \eta_l \Delta\tilde{M}_l)^{-1}}$ unless $\sum_{l=1}^{s} \eta_l \Delta\tilde{M}_l = 0$. However, we have already known that at least one $\Delta\tilde{M} \neq 0$ and η_l are randomly chosen, so $\sum_{l=1}^{s} \eta_l \Delta\tilde{M}_l = 0$ only with probability $1/q$, which is negligible.

Thus, the difference between $\mathcal{A}$'s probabilities of success in Game 0 and Game 1 is negligible.

Game 2. Game 2 is the same as Game 1, with one difference. $\mathcal{C}$ keeps a list of his responses to $HVT-Gen$ queries made by $\mathcal{A}$, and observes each response for $ProofGen$ executed by $\mathcal{A}$. If in any of these responses $\mathcal{A}$ is successful (i.e., $\mathcal{V}$ outputs VALID), but at least one of $\mathcal{A}$'s aggregated block sectors $\tilde{M}_l' \neq \gamma + \delta \sum_{(j,k)\in I} a_{jk}\tilde{m}_{jkl}$, $\mathcal{C}$ declares failure and aborts.

Analysis. We now show that if $\mathcal{A}$ causes $\mathcal{C}$ to abort with nonnegligible probability we can construct a simulator $\mathcal{S}$ that solves the DL problem. $\mathcal{S}$ is given as inputs values $g, h \in G$, its goal is to output θ which $h = g^{\theta}$. $\mathcal{S}$ behaves like $\mathcal{C}$ in Game 1, with the following differences:

1) In the responses to the $HVT-Gen$ queries, $\mathcal{S}$ randomly chooses $\alpha \in Z_q$, sets $v = g^{\alpha}$, randomly chooses $\{(\zeta_l, \eta_l) \mid l = 1, \ldots, s\}$, sets $\{u_l = g^{\zeta_l} h^{\eta_l} \mid l = 1, \ldots, s\}$, sets $\sigma = H_1(NID \parallel v \parallel u_1 \parallel \ldots \parallel u_s)^x$, randomly chooses $r_{jk}^{(NID)}$, programs the random oracle H_2 as $H_2(NID \parallel FID_j \parallel k) = g^{r_{jk}^{(NID)}} / (g^{\sum_{l=1}^{s} \zeta_l \tilde{m}_{jkl}} h^{\sum_{l=1}^{s} \eta_l \tilde{m}_{jkl}})$. We note that α is randomly chosen in Game 2.
2) $\mathcal{S}$ continues interacting with $\mathcal{A}$ until the condition specified in Game 2 occurs. The correct response satisfies the equation (3), and from the change made in Game 1 we know that $T' = T$, therefore we have:

$$\begin{aligned} & e((\prod\nolimits_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\delta} \\ & \cdot \prod\nolimits_{l=1}^{s} u_l{}^{\tilde{M}_l}, v) \\ = \ & \Gamma \cdot e(T, g)^{\delta} \\ = \ & \Gamma \cdot e(T', g)^{\delta} \\ = \ & e((\prod\nolimits_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\delta} \\ & \cdot \prod\nolimits_{l=1}^{s} u_l{}^{\tilde{M}_l'}, v). \end{aligned}$$

Then, we have $\prod_{l=1}^{s} u_l{}^{\tilde{M}_l} = \prod_{l=1}^{s} u_l{}^{\tilde{M}_l'}$, therefore:

$$\begin{aligned} 1 &= \prod\nolimits_{l=1}^{s} u_l{}^{\Delta\tilde{M}_l} \\ &= \prod\nolimits_{l=1}^{s} (g^{\zeta_l} h^{\eta_l})^{\Delta\tilde{M}_l} \\ &= g^{\sum_{l=1}^{s} \zeta_l \Delta\tilde{M}_l} h^{\sum_{l=1}^{s} \eta_l \Delta\tilde{M}_l}. \end{aligned}$$

We can see that we have found the solution the DL problem $\theta = -\sum_{l=1}^{s} \zeta_l \Delta\tilde{M}_l \cdot (\sum_{l=1}^{s} \eta_l \Delta\tilde{M}_l)^{-1}$, unless $\sum_{l=1}^{s} \eta_l \Delta\tilde{M}_l = 0$. But we have already known that at least one $\tilde{M}_l' \neq \gamma + \delta \sum_{(j,k)\in I} a_{jk}\tilde{m}_{jkl} = \tilde{M}_l$, therefore at least one $\Delta\tilde{M}_l \neq 0$. Because η_l are randomly chosen, so $\sum_{l=1}^{s} \eta_l \Delta\tilde{M}_l = 0$ with probability $1/q$, which is negligible.

Thus, the difference between $\mathcal{A}$'s probabilities of success in Game 1 and Game 2 is negligible.

Wrapping up. In Game 2, $\mathcal{A}$ is constrained from answering any verification query with values other than those that would have been computed by $\mathcal{C}$. We have already known that if the CDH problem is hard (implies the DL problem is hard) in G, there is only a negligible difference in the success probability of $\mathcal{A}$ in Game 2 compared to Game 0, where $\mathcal{A}$ is not constrained in this manner. Finally, if we select different coefficients a_{jk} (by setting different τ) on the same set of indices $I = (j, k)$, the same NID, the same set of $\{FID_j\}$, and the same set of $\{m_{jk}\}$, then execute $\#I$ times different challenges, we can compute all $\{\tilde{m}_{ijkl}\}$ by solving s linear equation groups, except possibly with negligible probability.

This completes the proof of Theorem 2. ■

C. Privacy-preserving

The privacy-preserving of IDLP-RDIC is guaranteed by Theorem 3.

Theorem 3 (Privacy-preserving): In IDLP-RDIC, if the CDH problem is hard in G, the verifier cannot derive any information about the values of $\{\tilde{m}_{ijkl}\}$ from the proof of integrity $P = (M, T, E, \Gamma)$ and even the $PoV = (R, \tau, P, Result, IDS_i(R), IDS_d(R, IDS_i(R), \tau), Sig_{RS}(P, ts), IDS_d(Result, ts))$ in the IDBCs.

Proof: The only two parts related to $\{\tilde{m}_{ijkl}\}$ in P and PoV are M and T. Now we prove that the verifier cannot derive any information about the values of $\{\tilde{m}_{ijkl}\}$ from M or T.

One the one hand, because γ is randomly chosen for each $ProofGen$, $\delta \sum_{(j,k)\in I} a_{jk}\tilde{m}_{jkl}$ is blinded by γ using a one-time-pad encryption. Even if the verifier knows $\Gamma = e(\prod_{l=1}^{s} u_l, v)^{\gamma}$ and $e(\prod_{l=1}^{s} u_l, v)$, γ is still hidden by the hardness of the DL problem. Thus, the verifier cannot derive any information about the values of $\{\tilde{m}_{ijkl}\}$ from M.

One the other hand, from:

$$\begin{aligned} & T \\ = \ & \prod\nolimits_{(j,k)\in I} ((H_2(NID \parallel FID_j \parallel k) \cdot \prod\nolimits_{l=1}^{s} u_l{}^{\tilde{m}_{jkl}})^{\alpha})^{a_{jk}} \\ = \ & (\prod\nolimits_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\alpha} \\ & \cdot (\prod\nolimits_{(j,k)\in I} \prod\nolimits_{l=1}^{s} u_l{}^{a_{jk}\tilde{m}_{jkl}})^{\alpha} \end{aligned}$$

we can see that $(\prod_{(j,k)\in I} \prod_{l=1}^{s} u_l{}^{a_{jk}\tilde{m}_{jkl}})^{\alpha}$ is blinded by $(\prod_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\alpha}$. However, the only way to compute $(\prod_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}})^{\alpha}$ for the verifier is to use g, g^{α}, and $\prod_{(j,k)\in I} H_2(NID \parallel FID_j \parallel k)^{a_{jk}}$, which is a CDH problem. Thus, the verifier cannot derive any information about the values of $\{\tilde{m}_{ijkl}\}$ from T.

This completes the proof of Theorem 3. ■

REFERENCES

[1] Gartner. Internet of Things (IoT). [Online]. Available: https://www.gartner.com/en/information-technology/glossary/internet-of-things. Accessed on: Aug. 20, 2020.

[2] M. Lippett. Fixing the biggest IoT issue – Data security. [Online]. Available: https://www.infosecurity-magazine.com/opinions/fixing-biggest-iot-data. Accessed on: Aug. 20, 2020.

[3] L. Abrams. (2019, Sept.) Amazon AWS outage shows data in the cloud is not always safe. [Online]. Available: https://www.bleepingcomputer.com/news/technology/amazon-aws-outage-shows-data-in-the-cloud-is-not-always-safe. Accessed on: Aug. 20, 2020.

[4] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conference on Computer and Communications Security - CCS*, 2007, pp. 598–609.

[5] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptol.*, vol. 26, 2013, pp. 442–483.

[6] B. Sengupta and S. Ruj, "Efficient proofs of retrievability with public verifiability for dynamic cloud storage," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 138–151, Jan./Mar. 2020.

[7] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.

[8] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A New Privacy-Aware Public Auditing Scheme for Cloud Data Sharing with Group Users," *IEEE Trans. Big Data*, to be published, doi: 10.1109/TBDATA.2017.2701347.

[9] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Trans. Depend. Sec. Comput.*, vol. 17, no. 3, pp. 608–619, May/June 2020.

[10] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.

[11] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1931–1940, Aug. 2017.

[12] H. Wang, "Identity-based distributed provable data possession in multicloud storage," *IEEE Trans. Serv. Comput.*, vol. 8, no. 2, pp. 328–340, Mar./Apr. 2015.

[13] Y. Yu, M. Au, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[14] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 72–83, Jan./Feb. 2019.

[15] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 331–346, Feb. 2019.

[16] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 824–835, Sept./Oct. 2019.

[17] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, "Privacy-preserving public auditing protocol for low-performance end device in cloud," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2572–2583, Nov. 2016.

[18] Y. Wang, Q. Wu, B. Qin, S. Tang, and W. Susilo, "Online/offline provable data possession," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 5, pp. 1182–1194, May 2017.

[19] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conference on Computer and Communications Security - CCS*, 2009, pp. 213–222.

[20] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, 2011, pp. 847–859.

[21] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and 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. Comput.*, vol. 64, no. 9, pp. 2609–2622, Sept. 2015.

[22] P. Nohe. (2018, Sept.) What is a rogue certificate? How do you prevent them? [Online]. Available: https://www.thesslstore.com/blog/what-is-a-rogue-certificate. Accessed on: Aug. 20, 2020.

[23] A. Shamir. "Identity-based cryptosystems and signature schemes," in *Proc. CRYPTO 84*, 1984, pp. 47–53.

[24] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," In *Proc. 7th International Conference on the Theory and Application of Cryptology and Information Security - ASIACRYPT*, 2001, pp. 514–532.

[25] S. Nakamoto. (2008, Oct.) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf. Accessed on: Aug. 20, 2020.

[26] A. Kaci and A. Rachedi, "PoolCoin: Toward a distributed trust model for miners' reputation management in blockchain," in *Proc. IEEE 17th Annual Consumer Communications & Networking Conference - CCNC*, 2020.

[27] Y. Yahiatene, A. Rachedi, M. A. Riahla, D. E. Menacer, and F. Nait-Abdesselam, "A blockchain-based framework to secure vehicular social networks," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 8, e3650, 2019.

[28] A. Kaci and A. Rachedi. "Toward a machine learning and software defined network approaches to manage miners' reputation in blockchain," *Journal of Network and Systems Management*, vol. 28, pp. 478–501, 2020.

[29] D. Galindo, J. Herranz, and E. Kiltz, "On the generic construction of identity-based signatures with additional properties," in *Proc. 12th International Conference on the Theory and Application of Cryptology and Information Security - ASIACRYPT*, 2006. pp. 178–193.

[30] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Compu.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

[31] M. A. Ferrag and L. Maglaras, "DeepCoin: A novel deep learning and blockchain-based energy exchange framework for smart grids," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1285–1297, Nov. 2020.

[32] A. D. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, "A decentralized privacy-preserving healthcare blockchain for IoT," *Sensors*, vol. 19, no. 326, 2019.

[33] A. Derhab, M. Guerroumi, A. Gumaei, L. Maglaras, M. A. Ferrag, M. Mukherjee, and F. A. Khan, "Blockchain and random subspace learning-based IDS for SDN-enabled industrial IoT security," *Sensors*, vol. 19, no. 3119, 2019.

[34] Verizon. (2020, Feb.) 5G speed: How fast is 5G? [Online]. Available: https://www.verizon.com/about/our-company/5g/5g-speed-how-fast-is-5g. Accessed on: Aug. 20, 2020.

[35] Verizon. (2020, Feb.) What is the Latency of 5G? [Online]. Available: https://www.verizon.com/about/our-company/5g/5g-latency. Accessed on: Aug. 20, 2020.

[36] B. Lynn. (2007, June) The pairing-based cryptography library (PBC). [Online]. Available: https://crypto.stanford.edu/pbc. Accessed on: Aug. 20, 2020.

[37] OpenSSL Software Foundation. (1999) OpenSSL: Cryptography and SSL/TLS toolkit. [Online]. Available: https://www.openssl.org. Accessed on: Aug. 20, 2020.

[38] J. C. Cha and J. H. Cheon, "An identity-based signature from gap Diffie-Hellman groups," In *Proc. 6th International Workshop on Practice and Theory in Public Key Cryptography - PKC*, 2003, pp. 18–30.

Su Peng is a lecturer at Shenyang Aerospace University, China. He received his Ph.D. degree in Computer Application Technology from the School of Computer Science and Engineering at Northeastern University, China, in 2019. His research interests include cryptography and blockchain.

Liang Zhao is an associate professor at Shenyang Aerospace University, China. He received his Ph.D. degree in Computing from the School of Computing at Edinburgh Napier University, United Kingdom, in 2011. Before joining Shenyang Aerospace University, he worked as an associate senior researcher in Hitachi (China) Research and Development Corporation from 2012 to 2014. His research interests include VANETs, SDN and MEC.

Ahmed Y. Al-Dubai is a professor of Networking and Communication Algorithms in the School of Computing at Edinburgh Napier University, United Kingdom. He received the Ph.D. degree in Computing from the University of Glasgow, United Kingdom, in 2004. His research interests include Communication Algorithms, Mobile Communication, Internet of Things, and Future Internet. He received several international awards. He is a senior member of IEEE.

Albert Y. Zomaya is a chair professor and director of the Center for Distributed and High Performance Computing at the University of Sydney, Australia. He has published more than 600 scientific papers and is an author, co-author, or editor of more than 20 books. He is the Editor-in-Chief of IEEE Transactions on Sustainable Computing and ACM Computing Surveys and serves as an Associate Editor for several leading journals. He is a Fellow of IEEE, AAAS, IET, and member of Academia European.

Jia Hu received the B.Eng. and M.Eng. degrees in electronic engineering from the Huazhong University of Science and Technology, China, in 2006 and 2004, respectively, and the Ph.D. degree in computer science from the University of Bradford, U.K., in 2010. He is currently a Senior Lecturer of computer science with the University of Exeter. His research interests include mobile and ubiquitous computing, resource optimization, applied machine learning, and network security.

Geyong Min is a professor of High-performance Computing and Networking in the Department of Mathematics and Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received his Ph.D. degree in computing science from the University of Glasgow, United Kingdom, in 2003, and his B.Sc. degree in computer science from Huazhong University of Science and Technology, China, in 1995. His research interests include future Internet, computer networks, wireless communications, multimedia systems, information security, high-performance computing, ubiquitous computing, modeling, and performance engineering.

Qiang Wang received his B.S. degree of Information Security and the M.S. degree of Software Engineering from Northeastern University, China, in 2014 and 2016, respectively. He is currently working toward the Ph.D. degree in Software College, Northeastern University, China. His research interests include verifiable computation and secure multi-party computation.