HUCDO: A Hybrid User-Centric Data Outsourcing Scheme

KE HUANG, University of Electronic Science and Technology of China
XIAOSONG ZHANG, University of Electronic Science and Technology of China
XIAOFEN WANG, University of Electronic Science and Technology of China
YI MU, Fujian Normal University
FATEMEH REZAEIBAGHA, Murdoch University
GUANGQUAN XU, Tianjin University
HAO WANG, Norwegian University of Science and Technology
XI ZHENG, Maquarie University
GUOMIN YANG, University of Wollongong
QI XIA, University of Electronic Science and Technology of China
XIAOJIANG DU, Temple University

Outsourcing helps relocate data from the cyber-physical system (CPS) for efficient storage at low cost. Current server-based outsourcing mainly focuses on the benefits of server, this cannot attract users well as their security, efficiency and economy are not guaranteed. To solve with this issue, a hybrid outsourcing model which exploits both cloud server and edge devices to store data is needed. Meanwhile, the requirements of security and efficiency are different under specific scenarios. There lacks a comprehensive solution to consider the above issues at one time for the sake of users. In this work, we overcome the above issues by proposing the first hybrid user-centric data outsourcing (HUCDO) scheme, it allows users to outsource data securely, efficiently and economically via different CPSs. Shortly, our contributions consist of theories, implementations and evaluations. Our theories include the first homomorphic collision-resistant chameleon hash (HCCH) and homomorphic designated-receiver signcryption (HDRS). As implementations, we instantiate how to use our proposals to outsource small or large-scale of data through distinct CPS respectively. Additionally, a blockchain with proof-of-discrete-logarithm (B-PoDL) is instantiated to help improve our performance. Lastly, as demonstrated by our evaluations, our proposals are secure, efficient and economic for users to implement while outsourcing their data via CPSs.

CCS Concepts: • **Security and Privacy** → **Cryptography**.

Additional Key Words and Phrases: chameleon hash, signcryption, hybrid data outsourcing, blockchain

1 INTRODUCTION

Cyber physical system (CPS) [1] seeks to integrate and coordinate physical and computational elements at a high level, which it paves the way to industry 4.0 [2] and develops modern system that operates in a more intelligent and reliable way. CPS shares basic infrastructure as internet-of-things (IoT) [3, 4]) did and it has largely driven by cloud computing techniques [5] in recent years.

Generally, data outsourcing allows for migration of data from the user device to cloud server to reclaim space from local-side. The server keeps data preserved and audited, and allows for remote access whenever it is required. However, since CPSs diverged to heterogeneous

Authors' addresses: Ke Huang, kh936@uowmail.edu.au, University of Electronic Science and Technology of China; Xiaosong Zhang, University of Electronic Science and Technology of China; Xiaofen Wang, University of Electronic Science and Technology of China; Yi Mu, Fujian Normal University; Fatemeh Rezaeibagha, Murdoch University; Guangquan Xu, Tianjin University; Hao Wang, Norwegian University of Science and Technology; Xi Zheng, Maquarie University; Guomin Yang, University of Wollongong; Qi Xia, University of Electronic Science and Technology of China; Xiaojiang Du, Temple University.

devices (personal computer, mobile phone and etc), their capabilities regarding storage and computing are different. Therefore, it calls for a smart solution to maximize the potential of each distinct CPS in heterogeneous outsourcing environment.

Early outsourcing schemes mainly focus on the gain-and-loss of server, which is reasonable when computing power and storage capacity are hard to come by. However, with a more insightful perspective of data value and gaining users [6], it is reasonable to consider the interests of users in order to prosper the outsourcing business. From a typical view, security, efficiency and economy are basic demands of users.

1.1 Problem Statement and Motivation

Data outsourcing schemes are facing security, efficiency and economy problems. We briefly state each issue and give our motivation behind.

For security [7], it is suggested to keep outsourced data under encryption for privacy concerns. Symmetric encryption [8] is acknowledged as an ideal way to encrypt massive data, but it leads to the key management problem. Public-key encryption overcomes key management issue but it costs too much at large scale of data. So, it calls for a comprehensive solution.

For efficiency, the server-centric cloud storage ignores the availability of edge devices for storing data [9]. With the booming of smart devices, there are many edge devices with considerable space to spare. Meanwhile, short-range transmission between these edge devices (e.g., blue tooth and etc [10]) wins long-distance transmission. If there is any edge devices nearby at disposal, a faster data outsourcing solution is available.

For economic reasons, users will more actively participate in outsourcing business if they are rewarded accordingly. In fact, there are already successful cases of data-driven businesses [6] where crypto-currencies are rewarded to users as incentives. As bitcoin and other blockchains [11] have been widely and trivially used nowadays, we omit to discuss the initiation or maintenance of matters. Instead, we only focus on the core technical details of launching the blockchain.

To address above problems, we are motivated to devise a hybrid and user-centric framework which exploits both cloud server, edge device and blockchain to maximize user's security, efficiency and economy.

1.2 Contributions

In this work, we propose a hybrid user-centric data outsourcing (HUCDO) scheme to ensure security, efficiency and economy of users who are engaged in outsourcing business via different cyber-physical systems (CPSs). Our contributions can be summarized on HUCDO as follows:

(1). As theoretical contributions, we propose the first homomorphic collision-resistant chameleon hash (HCCH) and homomorphic designated-receiver signcryption (HDRS). Our HCCH serves as a special hash function to help outsource large file, and our HDRS is used as privacy protection for small-scale of data during outsourcing.

(2). As practical implementations, we instantiate how to use our proposed HCCH and HDRS for outsourcing. Additionally, we instantiate the details of a blockchain with proof-of-discrete-logarithm (B-PoDL) to help improve the performance of our proposals.

(3). As evaluations, we conduct a comprehensive analysis of our proposals. The evidences show that our proposals are secure, efficient and economic for users to implement.

2 RELATED WORK

In this section, we study data outsourcing [12], chameleon hash (CH) [13] and signcryption [14] scheme. Basically, data outsourcing is the background of this work, CH and signcryption scheme are meant to implement outsourcing service efficiently and securely (as instantiated in section 6). The motivation behind each study is followed by.

2.1 Data Outsourcing

Data outsourcing is closely related to notions of provable data possession (PDP) [12] and data auditing. Informally, it allows the user to upload data to remote server for storage, and periodically audit it to ensure integrity. Ateniese et al. [12] proposed the first notion and security model of PDP where the user is allowed to verify data file without retrieving it. Then, Juels et al. [15] proposed the notion of proof-of-retrievability (PoR) to guarantee data retrieval by erasure code and random sampling. With the expansion of studies on data outsourcing, research regarding data dynamic operation [16], public auditability [17] and others have being considered. To ensure the efficiency during outsourcing, deduplication is a popular technique to consider [18], since it was used to delete redundant files at user-side to reclaim space. Since Douceur et al. [18] proposed notion of convergent encryption (CE) by the idea of hash-as-a-key, the research line expands to multiple aspects. Noticeably, for rigorous security, Bellare et al. [19] proposed message-locked encryption (MLE) which formally answered what security can CE and MLE achieve.

In this work, we mainly focus on how to process data securely for outsourcing. Specifically, we identify two different outsourcing scenarios where data is outsourced at different scales. We research on hashing and signcryption schemes, our proposals can help build secure and efficient outsourcing services as we will later instantiate in section 6.

2.2 Chameleon Hash

Informally, the chameleon hash is a trapdoor one-way hash function where finding the collision is hard without the trapdoor key. Krawczyk pand Rabin [13] first proposed chameleon signature in 2000. Later on, Ateniese et al. [20] extended it to the first identity-based chameleon hash while solving the key-exposure problem. Key-exposure problem was first identified by Ateniese et al. [21], in which, it is infeasible to extract trapdoor key from any given collisions. It was until the work of Camenisch et al. [22] which formally proved that collision-resistance is a stronger notion than key-exposure problem. In addition, they proposed chameleon hash with ephemeral trapdoors in [22] to prevent finding collisions without ephemeral trapdoor. In another work Krenn et al. [23] proposed chameleon-hashes with dual long-term trapdoors where the hashing party can choose between using a fresh second trapdoor and reusing an existing one.

We summarized relevant schemes in Table 1 as an overview. As it is shown in Table 1. most hash schemes do not involve in pairing-based computations (although there exists such scheme, however we do not instantiate them here), due to the hash function is generally designed to be practically fast. So, as long as the security is guaranteed, the chameleon hash function can serve as an efficient tool to process large scale of data. With smart design, it can breed many functions such as: sanitizable signature [28], public-key encryption [29] and etc. In this work, we specifically instantiate to use it for outsourcing large-scale of data.

Table 1. Overview of Chameleon Hash Schemes

Scheme	Identity -based	Pairing -based	Collision -Resistance	Formal Proof	Assumptions
ICHA[20]	Yes	No	Yes	No	q-SDHP
CHWK [24]	No	No	Yes	No	CDHP
OKEP[21]	No	No	Yes	No	Factoring and DLP
KEFC[25]	No	No	Yes	No	CDHP
CHET [22]	No	No	Yes	Yes	RSA and DLP
CHDL [23]	No	No	Yes	Yes	RSA
CIKE[26]	No	No	Yes	No	Factoring
ACCH[27]	No	No	Yes	Yes	Factoring

Each scheme is named by the abbreviation of titles.

2.3 Signcryption

Informally, signcryption is a notion where encryption and signature are applied during single operation, as a result, data confidentiality and authentication are achieved simultaneously.

For the first time, Zheng [14] first proposed a signcryption scheme based on ElGamal signature and encryption in 1997. Then, Baek et al [30] formally proved the confidentiality of Zheng's signcryption scheme [14] under a rigorous security model. Since then, studies on signcryption schemes have kept prospering.

Table 2. Overview of Signcryption Schemes

Scheme	Identity -based	Online/ Offline	Assumption	Security in Standard Model
IOSL[31]	Yes	Yes	q-SDHP and q-BDHIP	No
OOIB[32]	Yes	Yes	l-BDHI and l-SDHP	No
IBOO[33]	Yes	Yes	k-mBDHIP	No
EIOE[34]	Yes	Yes	k-CCA1	No
AISI[35]	Yes	No	CDHP and DBDH	Yes
SISS[36]	Yes	No	CDHP and DBDH	Yes
FSIS[37]	Yes	No	CDHP and DBDH	Yes
IBSS[38]	Yes	No	CDHP and DBDH	Yes
PSIB[39]	Yes	No	MBSDH and DMBDHI	No

Each scheme is named by the abbreviation of titles. Denote following notions as: DMBDHI: Decisional Modified Bilinear Diffie-Hellman Inversion problem; MBSDH: Modified Bilinear Strong Diffie-Hellman problem; k-CCA1: k-Collision Attack Assumption; q-BDHIP: q-Bilinear Diffie-Hellman Inversion problem; DBDH: Decisional Bilinear Diffie-Hellman problem; l-SDH: l-Strong Diffie-Hellman assumption; k-mBDHIP: Modified BDHI for k values.

Generally, most signcryption schemes fall into two categories: public-key infrastructure -based (PKI-based) [40–42] and Identity-based (ID-based) [31–39, 43–48].

Among the above ID-based signcryption schemes, Chen and Malone-Lee's work [43] is the most efficient one. For pairing-based signcryption schemes, Barreto et al's work [48] is the most efficient among others because they proposed both provably secure signcryption and signature. Works of [35–38] offer schemes with practical security and are provably secure in the standard model. To aim for faster performance, works of [31–34] focus on on-line/off-line feature which speed up performance by off-line pre-computations. Among them, the scheme proposed by Lai et al [34] is the most efficient one.

We list an overview of the current schemes in Table 2. As it is shown in Table 2, all schemes are under identity-based infrastructure (to resolve key-management problem in symmetric encryptions). Almost half of them are provably secure under the standard model (security is guaranteed without reliance on random oracle). Based on the above, we can conclude that signcryption is an idea primitive to serve as privacy protection for small scale of data (length of data is fixed and short).

3 DEFINITIONS

In this section, we give definitions and security requirements for our proposed HCCH and HDRS.

3.1 System Model of HUDCO

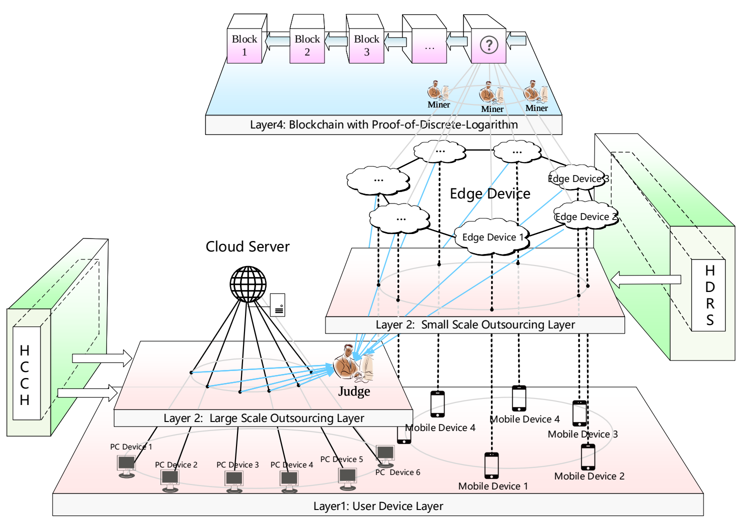


Fig. 1. The framework of hybrid user-centric data outsourcing (HUCDO) scheme

The framework of our hybrid user-centric data outsourcing (HUCDO) scheme is shown in Figure 1, it mainly consists of six parties explained as below.

User Device: Mobile or personal computer (PC) devices which outsource data from local-side to cloud server (or edge device). PC device is computationally powerful in computing whereas mobile device is computationally-limited.

Cloud Server: Conventional storage server with seemingly unlimited space and computing power due to economies of scale.

Edge Device: Devices which are geographically located at different places, and they have spare space to keep outsourced data for the nearest user device via a short-range transmission (e.g., blue-tooth).

Judge: A trusted third party delegated to settle any disputes (refer to HDRS.IntDeny in section 5.1) between user device and edge device regarding data validity.

B-PoDL: Blockchain with proof-of-discrete logarithm (as consensus) to help shift computational burdens from user device to miners (as instantiated by algorithm 1 and 2 in section

6.2). It is also used as an economic incentive for users to participate in outsourcing business more often.

Miner: A device equipped with powerful graphics processing units (GPUs), and it is utilized to maintain the blockchain and reach global consensus through competitive computations against each other ([11]).

As it is shown in Figure 1, our hybrid user-centric data outsourcing (HUCDO) scheme is captured by four layers. At layer 1, the user outsources their data from either pc device or mobile device. To outsource massive data, our proposed HCCH is applied to process data file (as instantiated by Figure 2 in section 6.1). In the opposite, when outsourcing data at a small scale (at layer 3), our proposed HDRS is used as privacy protection (as instantiated by Figure 3 in section 6.2). Additionally, a blockchain called B-PoDL at layer 4 is utilized to shift computations from the mobile device to miners, it also serves as an incentive mechanism for users.

3.2 Security Requirements of HCCH

A secure homomorphic collision-resistant chameleon hash (HCCH) should satisfy indistinguishability and public-collision resistance [22]. Correctness is obvious from inspection.

Experiment: $IND_{\mathcal{A}}^{\mathsf{HCCH}}(\lambda)$
$\mathsf{param}_{\mathsf{HCCH}}^{\mathsf{ch}} \leftarrow \mathsf{HCCH.Setup}(\lambda)$
$(hk_{ch}, tk_{ch}) \leftarrow \mathsf{HCCH.KeyGen}(\mathsf{param}_{\mathsf{HCCH}}^{\mathsf{ch}})$
$b \leftarrow \{0, 1\}$
$a \leftarrow$
$\mathcal{A}^{\mathsf{Hash\&Forge}(tk_{ch},\cdots,a),\mathsf{HCCH.Forge}(tk_{ch},\cdots)}(hk_{ch})$
where oracle $\mathsf{Hash\&Forge}$ on input
$(tk_{ch}, \mathsf{CID}, M, M^{'}, b)$:
Set $(\mathcal{H}, r) \leftarrow \mathsf{HCCH.Hash}(hk_{ch}, \mathsf{CID}, M, \alpha)$
Set $(\mathcal{H}^{'}, r^{'}) \leftarrow \mathsf{HCCH.Hash}(hk_{ch}, \mathsf{CID},$
$M^{'}, \alpha^{'})$
Set $(r^{''}) \leftarrow$
$\mathsf{HCCH.Forge}(\mathsf{tk_{ch}}, \mathsf{CID}, M, (M^{'}, \mathcal{H}, r^{'}))$
If $\mathsf{HCCH.Verify}(\mathsf{hk_{ch}}, \mathsf{CID}, M^{'}, \mathcal{H}^{'}, r^{'}) = \perp$
$\vee r'' = \perp$, return $\perp$
If $b = 0$, return $(\mathcal{H}, r)$
If $b = 1$, return $(\mathcal{H}', r'')$
return 1, if $a = b$;
else, return 0

Experiment: $PColRes_{\mathcal{A}}^{\mathsf{HCCH}}(\lambda)$
$\mathsf{param}_{\mathsf{HCCH}}^{\mathsf{ch}} \leftarrow \mathsf{HCCH.Setup}(\lambda)$
$(tk_{ch}, hk_{ch}) \leftarrow \mathsf{HCCH.KeyGen}(\mathsf{param}_{\mathsf{HCCH}}^{\mathsf{ch}})$
$(\mathsf{CID}^*, M^*, r^*, M^{**}, r^{**}, \mathcal{H}^*) \leftarrow$
$\mathcal{A}^{Forge'(tk_{ch},\cdots)}(hk_{ch})$
where oracle $Forge'$ on input
$(tk_{ch}, \mathsf{CID}, M, M', r, \mathcal{H})$:
Return $\perp$ if
$0 \leftarrow \mathsf{HCCH.Verify}(hk_{ch}, \mathsf{CID}, M, \mathcal{H}, r)$
$(r') \leftarrow$
$\mathsf{HCCH.Forge}(tk_{ch}, \mathsf{CID}, M', (M, \mathcal{H}, r))$
If $r' = \perp$, return $\perp$
$\mathcal{QL} \leftarrow \mathcal{QL} \cup \{\mathsf{CID}\}$ where $\mathcal{QL}$ denotes a list of query history
Return r'
Output 1 if $1 \leftarrow$
$\mathsf{HCCH.Verify}(\mathsf{hk_{ch}}, \mathsf{CID}^*, (\mathsf{M}^*, \mathcal{H}^*, \mathsf{r}^*)) = 1 \wedge$
$1 \leftarrow \mathsf{HCCH.Verify}(hk_{ch}, \mathsf{CID}^*, (M^{**},)$
$\mathcal{H}^*, r^{**}) \ \wedge \ \mathsf{CID}^* \notin \mathcal{QL} \ \wedge \ M^* \neq M^{**}$
else, return 0, if $a = b$

Indistinguishability: Indistinguishability can be achieved if no adversary should be able to efficiently distinguish the chameleon randomness r generated from HCCH.Hash or HCCH.Forge. Our proposed HCCH is indistinguishable if for any efficient adversary $\mathcal{A}$, the advantage in winning following experiment $IND_{\mathcal{A}}^{\mathsf{HCCH}}(\lambda)$ is negligible, i.e., Pr| $[IND_{\mathcal{A}}^{\mathsf{HCCH}}(\lambda) = 1] - \frac{1}{2} \; |\leq \nu(\lambda)$ where ν is a negligible function [22]. Additionally, the indistinguishability between randomness generated from HCCH.Int (homomorphic integration) and HCCH.Hash (or HCCH.Forge) can also be captured by the above experiment if we adapt the experiment slightly. Due to space limitations, we omit details.

Public Collision-resistance: Public Collision-resistance [22] requires that no adversary should be able to find any new collisions even if it is allowed to query a forging oracle Forge'. Obviously, the output should be fresh and has not been queried before. Our proposed HCCH is public collision-resistant if for any efficient adversary $\mathcal{A}$ against our HCCH, we have $\Pr[IND_{\mathcal{A}}^{\mathsf{HCCH}}(\lambda) = 1] \leq \nu(\lambda)$ where ν is a negligible function. In addition, collision resistance is a stronger notion than key-exposure freeness as noted by [22].

3.3 Security Requirements of HDRS

A secure homomorphic designated-receiver signcryption (HDRS) scheme should satisfy privacy, weak unforgeability, soundness of homomorphic integration and non-repudiation. Correctness is obvious from inspection.

Privacy: Our proposed HDRS is private if the encryption is indistinguishable against chosen-plaintext-attacks (IND-CPA). This is captured by experiment $IND-CPA_{\mathcal{A}}^{\mathsf{HDRS}}(\lambda)$. Assume there exists an efficient adversary $\mathcal{A}$ against our scheme, our HDRS is private if $\Pr|\ [IND-CPA_{\mathcal{A}}^{\mathsf{HDRS}}(\lambda) = 1] - \frac{1}{2}\ |\leq \nu(\lambda)$ holds where ν is a negligible function.

Experiment: $IND-CPA_{\mathcal{A}}^{\lambda}$
$\mathsf{param}_{\mathsf{HDRS}}^{\mathsf{ch}} \leftarrow \mathsf{HDRS.Setup}(\lambda)$
$(pk_R, sk_R)(pk_S, sk_S) \leftarrow \mathsf{HDRS.KeyGen}$
$(\mathsf{param}_{\mathsf{HCCH}}^{\mathsf{ch}})$
$b^* \leftarrow \mathcal{A}^{\mathsf{Encrypt}}(pk_R, pk_S, c_b, m_0, m_1)$
where oracle $\mathsf{Encrypt}$ on input:
$pk_R, \mathsf{CID}, m, \alpha$:
return $C = g^m (h \cdot pk_R)^{\alpha}$, i.e.,
the same way as $\mathsf{HDRS.Signcrypt}$ did for c_2
in a signcrypted message $c = (c_0, c_1, c_2, c_3)$
Here, $C_b = g^{m_b} (h \cdot pk_R)^{\alpha}$
where b is chosen randomly from $\{0, 1\}$
beside of $\mathcal{A}$'s view and α is chosen
randomly from Z_q and $|m_0| = |m_1|$
Return 1 if $b = b^*$
else, return 0

Experiment: $WUF_{\mathcal{A}}^{\lambda}$
$\mathsf{param}_{\mathsf{HDRS}}^{\mathsf{ch}} \leftarrow \mathsf{HDRS.Setup}(\lambda)$
$(pk_R, sk_R) \leftarrow \mathsf{HDRS.KeyGen}$
$(m^*, \mathsf{CID}^*, c^*) \leftarrow \mathcal{A}(pk_R, \mathsf{param}_{\mathsf{HDRS}}^{\mathsf{ch}})$
Return 1 if
$1 \leftarrow \mathsf{HDRS.Verify}(m^*, \mathsf{CID}^*, c^*, sk_R)$
otherwise, return 0.

Weak Unforgeability (WUF): Our proposed HDRS is unforgeable against weak chosen-message attacks (WUF) if no efficient adversary $\mathcal{A}$ can win experiment $WUF_{\mathcal{A}}^{\mathsf{HDRS}}(\lambda)$ defined as below with non-negligible advantage, i.e., $\Pr[EUF - CMA_{\mathcal{A}}^{\lambda} = 1] \leq \nu(\lambda)$ where ν is a negligible function. The unforgeability of our re-signature (generated by $\mathsf{HDRS.Re\text{-}Sign}$) follows definition of weak unforgeability as well, details are omitted.

Soundness of Homomorphic Integration: Our proposed HDRS satisfies soundness of homomorphic integration if no adversary $\mathcal{A}$ could successfully forge a proof to pass the verification without actually performing homomorphic integration. This is captured by experiment $HomInt_{\mathcal{A}}^{\lambda}$. We requires that $\Pr[HomInt_{\mathcal{A}}^{\lambda} = 1] \leq \nu(\lambda)$ holds for any efficient adversary $\mathcal{A}$ against our scheme where ν is a negligible function.

Non-repudiation: Our proposed HDRS satisfies non-repudiation if no adversary could deny a legitimate ciphertext signcrypted by himself. In other words, he cannot maliciously accuse the receiver of performing any non-negotiated actions. This is captured by the experiment of $NR_{\mathcal{A}}^{\lambda}$. We asks that $\Pr[NR_{\mathcal{A}}^{\lambda} = 1] \leq \nu(\lambda)$ holds for any efficient adversary $\mathcal{A}$ against our scheme where ν is a negligible function.

Experiment: $HomInt_{\mathcal{A}}^{\lambda}$
$\mathsf{param}_{\mathsf{HDRS}}^{\mathsf{ch}} \leftarrow \mathsf{HDRS.Setup}(\lambda)$
$(pk_R, sk_R)(pk_S, sk_S) \leftarrow \mathsf{HDRS.KeyGen}$
Given m, CID and c where
$c \leftarrow \mathsf{HDRS.Signcrypt}(\mathsf{CID}, m, sk_S, pk_R)$
and $1 \leftarrow \mathsf{HDRS.Verify}(m, \mathsf{CID}, c, sk_R)$
For ease of analysis, we limit number of integrated messages n to 1, i.e., $c = \{c_0, c_1, c_2, c_3\}$.
On a challenge $chal \leftarrow \mathsf{HDRS.IntChal}()$ for c
$\mathsf{proof}^* \leftarrow \mathcal{A}^{\mathsf{chal}}(pk_R, pk_S)$
Return 1 to signify success if
$1 \leftarrow \mathsf{HDRS.IntVerify}(chal, \mathsf{proof}^*, \mathsf{pk_R})$
else, return 0 as a failure

Experiment: $NR_{\mathcal{A}}^{\lambda}$
$\mathsf{param}_{\mathsf{HDRS}}^{\mathsf{ch}} \leftarrow \mathsf{HDRS.Setup}(\lambda)$
$(pk_R, sk_R)(pk_S, sk_S) \leftarrow \mathsf{HDRS.KeyGen}$
On given a customized identity CID and
$c \leftarrow \mathsf{HDRS.Signcrypt}(\mathsf{CID}, m, sk_S, pk_R)$
$(\alpha^{**}, m^{**}) \leftarrow \mathcal{A}^{\mathsf{Signcrypt}(sk_S, \cdots)}(c, \mathsf{CID})$
Here, oracle Signcrypt responds queries the same way as algorithm HDRS.Signcrypt did for each distinct m_i where $1 \leq i \leq q_s$
Return 1 if $0 \leftarrow \mathsf{HDRS.IntDeny}(c, \mathsf{CID}, (\alpha^{**}, m^{**}), pk_S, pk_R) \wedge c \leftarrow \mathsf{HDRS.Verify}(m, \mathsf{CID}, c, sk_R) \wedge m \neq m^*$
else, return 0.

4 THE PROPOSED HCCH AND SECURITY ANALYSIS

In this section, we give concrete construction of HCCH in the Gap Diffie-Hellman (GDH) group and security analysis based on definitions given in section 3.2. A Gap Diffie-Hellman (GDH) group is a group where CDHP is hard and DDHP is easy on it, its construction can be found in [49].

Additionally, our HCCH yields a chameleon signature in the form of $(m, r, SIGN(\mathcal{H}))$ where $SIGN()$ is a non-specified public-key signing scheme. Due to space limitations, further discussions are omitted.

4.1 Construction of HCCH Scheme

HCCH.Setup$(\lambda) \rightarrow (\mathsf{param}_{\mathsf{HCCH}})$: On input a security parameter λ, choose a GDH group G_1 with a generator g and prime order p. Set $H_1 : \{0,1\}^* \rightarrow Z_p$. Output system parameter $\mathsf{param}_{\mathsf{HCCH}} = \{G_1, p, g, H_1\}$.

HCCH.KeyGen$(\mathsf{param}_{\mathsf{HCCH}}) \rightarrow (tk, hk)$: On input system parameters $\mathsf{param}_{\mathsf{HCCH}}$, the algorithm randomly selects and integer $x \xleftarrow{R} Z_p^*$ as trapdoor key tk and computes $hk = y = g^x$ as hash key. Output trapdoor key and hash key (tk, hk).

HCCH.Hash$(hk, \mathsf{CID}, M, \alpha) \rightarrow (\mathcal{H}, r)$: On input a hash key hk, a customized identity CID, a message of arbitrary length $M \in \{0,1\}^*$ and a randomness $\alpha \xleftarrow{R} Z_p^*$, the HCCH.Hash algorithm computes $e = H_1(\mathsf{CID}, y)$ and $h = g^e$, and then computes chameleon randomness $r = (g^\alpha, y^\alpha)$ and chameleon hash $\mathcal{H} = g^{H_1(M)}(h \cdot y)^\alpha$. Output $(\mathcal{H}, r)$.

HCCH.Verify$(hk, \mathsf{CID}, (M, \mathcal{H}, r)) \rightarrow (0 \text{ or } 1)$: On input a hash key hk, a customized identity CID, a tuple (M, h, r) which includes a message $M \in \{0,1\}^*$, a chameleon hash $\mathcal{H}$ and a chameleon randomness $r = (g^\alpha, y^\alpha)$, the HCCH.Hash algorithm computes $e = H_1(\mathsf{CID}, y)$ and checks whether $< g, g^\alpha, y, y^\alpha >$ and $< g, g^\alpha, h \cdot y, \frac{\mathcal{H}}{g^{H_1(M)}} >$ are both valid Diffie-Hellman tuples. If both are, output 0; otherwise, output 1.

HCCH.Forge$(tk, \mathsf{CID}, M^{'}, (M, \mathcal{H}, r)) \rightarrow (r^{'} \text{ or } \perp)$: On input a trapdoor key tk, a customized identity CID, a new message $M^{'} \in \{0,1\}^*$, a tuple $(M, \mathcal{H}, r)$ which includes an original message $M \in \{0,1\}^*$, corresponding chameleon hash $\mathcal{H}$ and chameleon randomness r, the algorithm first runs $\mathsf{HCCH.Forge}(hk, \mathsf{CID}, (M, \mathcal{H}, r))$. If it is 0, outputs $\perp$; otherwise, the algorithm computes $e = H_1(\mathsf{CID}, y)$ and the new randomness $r^{'} = (g^{\alpha^{'}}, y^{\alpha^{'}}) = (g^\alpha \cdot g^{(H_1(M) - H_1(M^{'}))(x+e)^{-1}}, y^\alpha \cdot y^{(H_1(M) - H_1(M^{'}))(x+e)^{-1}})$. Then, the algorithm runs

HCCH.Verify$(hk, \mathsf{CID}, (H_1(M^{'}), \mathcal{H}, r^{'}))$ to check correctness. If it is 0, output $\perp$; otherwise, output $r^{'}$. Note that the forgery succeeds if equation 1 holds and both $(M, \mathcal{H}, r)$ and $(H_1(M^{'}), \mathcal{H}, r^{'})$ pass verifications of HCCH.Verify.

$$g^{H_1(M)}(h \cdot y)^{\alpha} = g^{H_1(M^{'})}(h \cdot y)^{\alpha^{'}} \tag{1}$$

HCCH.Int$(hk, (H_1(M_1), \mathcal{H}_1, r_1), \cdots, (H_1(M_n), \mathcal{H}_n, r_n), \mathsf{CID}) \to ((\tilde{M}, \tilde{\mathcal{H}}, \tilde{r})$ or $\perp)$:

On input a hash key hk, n tuples of $(H_1(M_i), \mathcal{H}_i, r_i)$ for $(1 \leq i \leq n)$ under same customized identity CID, the algorithm computes $\tilde{M} = \sum_{i=1}^{n} H_1(M_i)$, $\tilde{\mathcal{H}} = \prod_{i=1}^{n} \mathcal{H}_i$ and $\tilde{r} = (g^{\alpha}, y^{\alpha}) = (g^{\sum_{i=1}^{n} \alpha_i}, y^{\sum_{i=1}^{n} \alpha_i})$ where $\alpha = \sum_{i=1}^{n} \alpha_i$. Then, check whether $< g, g^{\alpha}, y, y^{\alpha} >$ and $< g, g^{\alpha}, h \cdot y, \frac{\tilde{\mathcal{H}}}{g^{H_1(\tilde{M})}} >$ are both valid Diffie-Hellman tuples. If yes, output $(\tilde{M}, \tilde{\mathcal{H}}, \tilde{r})$ as a homomorphic integration result; otherwise, output $\perp$. Since $\tilde{\mathcal{H}} = g^{\sum_{i=1}^{n} H_1(M_i)}(h \cdot y)^{\sum_{i=1}^{n} \alpha_i}$ where $\sum_{i=1}^{n} H_1(M_i) \in Z_p$, the result $(\tilde{\mathcal{H}}, \tilde{r})$ has the same form of the output of HCCH.Hash.

4.2 Security Analysis of HCCH

Indistinguishability:: We prove by game hopping as follows:

- **Game 0:** This is the original indistinguishability game where $b = 0$.
- **Game 1:** The same as Game 0 except hashing directly to derive $\mathcal{H}$.
- **Game 2:**: The same as Game 1 except hashing directly to derive $\mathcal{H}'$.

Denote E_i as the event of **Game i** won by $\mathcal{A}$ (i.e. $1 \leftarrow IND_{\mathcal{A}}^{\mathsf{HCCH}}(\lambda)$). Set **Game 0** as the original game defined in $IND_{\mathcal{A}}^{\mathsf{HCCH}}(\lambda)$, the advantage of $\mathcal{A}$ in winning **Game 0** is $Adv_{\mathcal{A}}^{IND} = |Pr[E_0] - \frac{1}{2}|$.

Transition from Game 0 to game 1: This hop only modifies the view of adversary $\mathcal{A}$ negligibly due to the indistinguishability of our HCCH. Otherwise, if $\mathcal{A}$ can distinguish this hop, an adversary $\mathcal{B}$ can be constructed to break the indistinguishability of HCCH. Concretely, $\mathcal{B}$ replaces hk_{ch0} with hk_{ch1} and queries oracle Hash&Forge to derive $\mathcal{H}$. Then, $\mathcal{B}$ relays the output from $\mathcal{A}$. Due to the indistinguishability of our HCCH, we have $|Pr[E_0] - Pr[E_1]| \leq \nu(\lambda)$.

Transition from Game 1 to game 2: This hop only modifies the view of adversary $\mathcal{A}$ negligibly due to the indistinguishability of our HCCH. Otherwise, if $\mathcal{A}$ can distinguish this hop, an adversary $\mathcal{B}$ can be constructed to break the indistinguishability of HCCH. Concretely, $\mathcal{B}$ replaces hk_{ch1} with hk_{ch2} and applies Hash&Forge to derive $\mathcal{H}'$. Then, $\mathcal{B}$ relays the output from $\mathcal{A}$. Analogically, we have $|Pr[E_1] - Pr[E_2]| \leq \nu(\lambda)$.

Last, for $b = 1$ we have $Pr[E_2] = \frac{1}{2}$. Based on the above, we can deduce that $Adv_{\mathcal{A}}^{IND} \leq Adv_{\mathcal{B}}^{IND} \leq \nu(\lambda)$. Since each hop modifies the view slightly and this change is beyond the adversary $\mathcal{A}$'s view, our HCCH is indistinguishable.

Public Collision-Resistance: Suppose $\mathcal{A}$ is an efficient adversary who breaks the indistinguishability of our HCCH, we briefly show how to construct a probabilistic polynomial time (PPT) algorithm $\mathcal{B}$ to solve q-SDHP [50]. On given a q-SDHP instance $(g, g^x, \cdots, g^{x^q})$, we denote it as $(A_0, A_1, \cdots, A_q)$ where $A_i = g^{x_i} \in G_1$ for $i = 1, \cdots, q$ and $A_0 = g$. Here, $x \in Z_p^*$ is unknown. In order to derive an answer $(c, g^{\frac{1}{(x+c)}})$ for some $c \in Z_q^*$ (which can either be designated [50] or not, for ease of analysis, we do not designate it). $\mathcal{B}$ interacts with $\mathcal{A}$ to derive an answer for q-SDHP as below:

Query: Adversary $\mathcal{A}$ issues q_s distinct queries $\{\mathsf{CID}_i, M_i^{'}, (M_i, \mathcal{H}, r_i)\}_{i \in [1, q_s]}$ under same hash key $hk = y$. Assume $q_s = q - 1$.

Response: For each M_i for $1 \leq i \leq q_s$, $\mathcal{B}$ generates corresponding response as follows: Set polynomial $f(z) = \prod_{i=1}^{q_s}(z+e_i) = \sum_{i=0}^{q_s} a_i z^i$ where $a_0, \cdots, a_{q_s}$ are coefficients of polynomial $f(z)$ and $e_i = H_1(\mathsf{CID}_i, y)$, $hk = y$ is the hash key. Define:

$$g' = \prod_{i=0}^{q_s}(A_i)^{a_i} = g^{f(z)} \text{ and } \tilde{h} = \prod_{i=1}^{q_s}(A_i)^{a_{i-1}} = g^{zf(z)} = {g'}^z$$

Next, we define polynomial $f_i(z) = f(z)/(z+e_i) = \prod_{j=1, j\neq i}^{q_s}(z+e_j)$ and $f_i(z) = \sum_{j=0}^{q_s-1}(b_j z^j)$. Then, $\mathcal{B}$ computes:

$$r_i^{'} = (g^{\alpha_i} \cdot {s_i}^{H_1(M_i)-H_1(M_i^{'})}, y^{\alpha_i} \cdot {s_i}^{x[H_1(M_i)-H_1(M_i^{'})]})$$

$$s_i = \prod_{j=0}^{q_s-1}(A_j)^{b_j} = (g')^{1/(x+e_i)} \text{ where } e_i = H_1(\mathsf{CID}_i, y)$$

Since equation holds for $g^{H_1(M_i)}(h_i \cdot y)^{\alpha_i} = g^{H_1(M_i')}(h_i \cdot y)^{\alpha_i^{'}}$ where $h_i = g^{e_i} = g^{H_1(\mathsf{CID}_i, y)}$, r' is the correct randomness to satisfy collision under customized identity CID_i and public key $(g', \cdots)$. Algorithm $\mathcal{B}$ replies adversary $\mathcal{A}$ with a list of q_s new randomness $(r_1', \cdots, r_{q_s}')$

Output: Adversary $\mathcal{A}$ wins by output $(\mathsf{CID}^*, M^*, r^*, M^{**}, r^{**}, \mathcal{H}^*)$ where $(M^*, \mathcal{H}^*, r^*)$ and $(M^{**}, \mathcal{H}^*, r^{**})$ are a collision and M^{**} has never been queried during query stage such that $g^{H_1(M^*)}(h^* \cdot y)^{\alpha^*} = g^{H_1(M^{**})}(h^* \cdot y)^{\alpha_i^{**}}$ where $h^* = g^{e^*} = g^{H_1(\mathsf{CID}^*, y)}$. We have:

$$r^{**} = (g^{\alpha^{**}}, y^{\alpha^{**}}) = (g^{\alpha^*} \cdot {s^*}^{H_1(M^*)-H_1(M^{**})}, y^{\alpha^*} \cdot {s^*}^{x\{H_1(M^*)-H_1(M^{**})\}})$$

$$s^* = (\frac{g^{\alpha^{**}}}{g^{\alpha^*}})^{\frac{1}{H_1(M^* - H_1(M^{**}))}} = (g')^{1/(x+e^*)} = g^{f(x)/(x+e^*)}$$

where $hk = x$ denotes the trapdoor key. We can parse f as $f(z) = \gamma(z)(z+e^*) + \gamma_{-1}$ for some $\gamma(y) = \sum_{i=0}^{q_s-1}\gamma_i z^i$ and $\gamma_{-1} \in Z_p$. Then, we can deduce by:

$$f(z)/(z+e^*) = \frac{\gamma_{-1}}{z+e^*} + \sum_{i=0}^{q_s-1}\gamma_i z^i$$

Since $\gamma_{-1} \neq 0$ and CID^* has never been queried before (i.e., $\mathsf{CID}^* \notin \{\mathsf{CID}_1, \cdots, \mathsf{CID}_{q_s}\}$), $(z+e^*)$ cannot divide $f(z)$. So, algorithm $\mathcal{B}$ calculates:

$$\pi = (s^* \cdot \sum_{i=1}^{q_s}(A_i)^{-\gamma_i})^{\frac{1}{\gamma_{-1}}} = g^{\frac{1}{x+e^*}}$$

and outputs (e^*, π) where $e^* = H_1(\mathsf{CID}^*, y)$ as an answer to the q-SDHP instance $(g, g^x, \cdots, g^{x^q})$. To bound the advantage of algorithm $\mathcal{B}$ which solves q-SDHP by using $\mathcal{A}$, ideas can be followed by [50], details are omitted.

5 THE PROPOSED HDRS AND SECURITY ANALYSIS

In this section, we give concrete construction of HDRS in the non-GDH groups and security analysis based on definitions given in section 3.3.

5.1 Construction of HDRS Scheme

HDRS.Setup(λ) $\rightarrow$ ($\mathsf{param}_{\mathsf{HDRS}}$): On input a security parameter λ, choose a group G_2 generated by g of prime order p. Set $H_2 : \{0,1\}^* \rightarrow Z_p$. Output system parameter $\mathsf{param}_{\mathsf{HDRS}} = \{G_2, p, g, H_2\}$.

HDRS.KeyGen($\mathsf{param}_{\mathsf{HDRS}}$) $\rightarrow$ (sk_{user}, pk_{user}): On input system parameters $\mathsf{param}_{\mathsf{HDRS}}$, the algorithm randomly selects three integers from Z_p as private key $sk_{user} = (x_{0,user}, x_{1,user}, x_{2,user})$ and computes $pk_{user} = (y_{0,user} = g^{x_{0,user}}, y_{1,user} = g^{x_{1,user}}$, $y_{2,user} = g^{x_{2,user}})$ as public key for a user. Concretely, $x_{0,user}$ is for de-signcryption, $x_{1,user}$ is for signing and $x_{2,user}$ is for verification of the signature scheme. Output (sk_{user}, pk_{user}).

HDRS.RKeyGen($\mathsf{param}_{\mathsf{HDRS}}, sk_{S_A}, sk_{S_B}$) $\rightarrow$ (k_{AB}): On input system parameters $\mathsf{param}_{\mathsf{HDRS}}$, two private keys x_{1,S_A} and x_{1,S_B} for user S_A and user S_B respectively. Generate a proxy re-signature key k_{AB} as follows: (1). The proxy P picks a random number $s \in Z_p$ and sends it to S_A, (2) S_A generates and sends $SIGN_{x_{1,A}}(\frac{s}{x_{1,A}})$ to S_B (we apply BLS signature [49] as SIGN() here), (3). S_B generates and sends $SIGN_{x_{1,A}}(\frac{s \cdot x_{1,B}}{x_{1,A}})$. Output a re-encryption key k_{AB}.

HDRS.Signcrypt($\mathsf{CID}, m, sk_S, pk_R$) $\rightarrow$ ($c, \perp$): On input a customized identity $\mathsf{CID} \in \{0,1\}^*$, a message $m \in Z_p$, private key sk_S of sender S, public key pk_R of receiver R, the algorithm first computes $e = H_2(\mathsf{CID}, pk_R)$ and $h = g^e$. Then, it selects a random number $\alpha \xleftarrow{R} Z_p^*$ and compute $c = (c_0.c_1, c_2, c_3)$ as follows:

$$c_0 = g^{\alpha}$$
$$c_1 = y_{1,S}^{\alpha}$$
$$c_2 = g^m (h \cdot y_{0,R})^{\alpha}$$
$$c_3 = (y_{1,R}^m \cdot y_{2,R}^{\alpha})^{x_{1,S}}$$

Output $c = (c_0.c_1, c_2, c_3)$ as message signcrypted by sender S for receiver R.

HDRS.Re-Sign(c_{S_A}, k_{AB}) $\rightarrow$ (c_{S_B}): On input a signcrypted message $c_{S_A} = (c_{0,S_A}, c_{1,S_A}, c_{2,S_A}, c_{3,S_A})$ signcrypted by sender S_A for receiver R, a re-signature key k_{AB}, the algorithm proceeds as: (1). Set $c_{0,S_B} = c_{1,S_A}^{k_{AB}}$ and $c_{2,S_B} = c_{1,S_A}^{k_{AB}}$, (2). Compute $c_{1,S_B} = c_{0,S_A}$ and $c_{3,S_B} = c_{3,S_A}^{k_{AB}}$. Output $c_{S_B} = (c_{0,S_B}, c_{1,S_B}, c_{2,S_B}, c_{3,S_B})$ as a message signcrypted by sender S_B to receiver R.

HDRS.De-Signcrypt(c, CID, sk_R) $\rightarrow$ (m): On input a signcrypted message $c = (c_0, c_1, c_2, c_3)$, a customized identity CID and private key sk_R of receiver R, the algorithm computes $e = H_2(\mathsf{CID}, pk_R)$ and de-signcrypts as equation 2:

$$m = \log_g^{\frac{c_2}{c_0^{(e+x_{0,R})}}} \tag{2}$$

HDRS.Verify(m, CID, c, sk_R) $\rightarrow$ (0 or 1): On input a plaintext $m \in Z_p$, a customized identity CID, a signcrypted message $c = (c_0, c_1, c_2, c_3)$ and private key $sk_R = \{x_{0,R}, x_{1,R}, x_{2,R}\}$ of receiver R, the algorithm checks whether equation 3 holds:

$$c_3 = y_{1,S}^{m \cdot x_{1,R}} \cdot c_1^{x_{2,R}} \tag{3}$$

If equation 3 holds, output 1; otherwise, output 0.

HDRS.IntChal() $\rightarrow$ ($chal$). No input, the sender S generates an order $P = \{i_j\}_{1 \le j \le n}$ to indicate a homomorphic integration of n messages as $\tilde{m} = \sum_{j=1}^{n} m_{i_j}$ under same customized identity CID. Then, the sender randomly generates a message $\tilde{m}' \in Z_p$. The algorithm outputs $chal = (P, \tilde{m}', \mathsf{CID})$ as a challenge for a homomorphic integration.

HDRS.Int&Prove$(chal, c, sk_R) \rightarrow (\tilde{c}, proof$ or $\perp)$: On input a challenge $chal = (P, \tilde{m}^{'}, \mathsf{CID})$, a set of n ciphertexts $c = \{c_{0,i_j}, c_{1,i_j}, c_{2,i_j}, c_{3,i_j}\}_{1 \leq j \leq n}$ generated from same sender S to same receiver R, and private key sk_R of receiver R, R first performs homomorphic integration based on challenge $chal$ to derive $\tilde{c}$ by:

$$\tilde{c_0} = \prod_{j=1}^{n} c_{0,i_j} = g^{\sum_{j=1}^{n} \alpha_{i_j}}$$

$$\tilde{c_1} = \prod_{j=1}^{n} c_{1,i_j} = y_{1,S}^{\sum_{j=1}^{n} \alpha_{i_j}}$$

$$\tilde{c_2} = \prod_{j=1}^{n} c_{2,i_j} = g^{\sum_{j=1}^{n} m_{i_j}} (h \cdot y_{0,R})^{\sum_{j=1}^{n} \alpha_{i_j}}$$

$$\tilde{c_3} = \prod_{j=1}^{n} c_{3,i_j} = (y_{1,R}^{\sum_{i_j=1}^{n} m_{i_j}} y_{2,R}^{\sum_{j=1}^{n} \alpha_{i_j}})^{x_{1,S}}$$

Denote the result as $\tilde{c} = (\tilde{c_0}, \tilde{c_1}, \tilde{c_2}, \tilde{c_3})$. Then, R runs $\mathsf{HDRS.De\text{-}Signcrypt}(\tilde{c}, \mathsf{CID}, sk_R)$ to derive plaintext $\tilde{m}$ where $\tilde{m} = \sum_{j=1}^{n} m_{i_j}$. Then, R runs $\mathsf{HDRS.Verify}(\tilde{m}, \mathsf{CID}, c, sk_R)$ for verification, if output 0, return $\perp$ and terminate; otherwise, R computes a forgery by:

$$\tilde{c_0}^{'} = g^{\alpha^{'}} = g^{\alpha} \cdot g^{(\tilde{m} - \tilde{m}^{'})(x_R + e)^{-1}}$$

Next, R computes $Z_1 = g^{\tilde{m}}$, $Z_2 = g^{\alpha}$, $Z_3 = g^{\alpha^{'}}$, $Z_4 = (\tilde{c_0})^{x_R} = (g^{\alpha})^{x_R}$ and $Z_5 = (\tilde{c_0}^{'})^{x_R} = (g^{\alpha^{'}})^{x_R}$. Finally, R computes $proof = \mathsf{SIGN}_{sk_R}(Z_1 \parallel Z_2 \parallel Z_3 \parallel Z_4 \parallel Z_5)$ as a proof for challenge $chal$. Output $(\tilde{c}, proof)$.

HDRS.IntVerify$(chal, proof, pk_R) \rightarrow (0$ or $1)$: On input a challenge $chal = (P, \tilde{m}^{'}, \mathsf{CID})$, a proof $proof = \mathsf{SIGN}_{sk_R}(Z_1 \parallel Z_2 \parallel Z_3 \parallel Z_4 \parallel Z_5)$, first verifies the validity of signature $\mathsf{SIGN}_{sk_R}()$ with pk_R, if it does not hold, return 0 and terminate; otherwise, compute $Z_6 = g^{\tilde{m}^{'}}$ and $e = H_2(\mathsf{CID}, pk_R)$. Then, checks whether equation 4 holds:

$$Z_1 \cdot {Z_2}^{e} \cdot Z_4 = Z_6 \cdot {Z_3}^{e} \cdot Z_5 \tag{4}$$

If equation 4 holds, the algorithm outputs 1; otherwise, output 0.

HDRS.IntDeny$(c^*, \mathsf{CID}, (\alpha, m), pk_S, pk_R) \rightarrow (0$ or $1)$: On input a dispute aggregated ciphertext $c^* = (c_0^*, c_1^*, c_2^*, c_3^*)$ from $\mathsf{HDRS.Int\&Prove}$ and corresponding customized identity CID, an evidence (α, m) which is revealed by sender S to deny the legitimacy of c^* where α denotes the original randomness chosen to compute c^*, and public keys pk_S and pk_R for sender S and receiver R respectively. A trusted judge can be employed to decide the legitimacy of received evidences (i.e., where c^* is forged by receiver R or not). The algorithm computes $h = g^e = g^{H_2(\mathsf{CID}, pk_R)}$ and checks whether : $c_2^* = g^m (h \cdot y_{0,R})^{\alpha}$ and $c_0^* \neq g^{\alpha}$. If both hold, return 1 to suggest that dispute ciphertext c^* is not a valid homomorphic integration result from $\mathsf{HDRS.Int\&Prove}$ as negotiated with sender R; otherwise, c^* is valid.

p

5.2 Security Analysis of HDRS

Privacy: Assuming $\mathcal{A}$ is a PPT adversary $\mathcal{A}$ who can break the IND-CPA security of our HDRS. We first prove by game hopping where each hop only changes $\mathcal{A}$'s view negligibly. Then, we can construct an algorithm $\mathcal{B}$ to use $\mathcal{A}$ (supposing he can distinguish between hops) to solve DDHP [51]. Accordingly, we can bound their advantages based on the above. First, we give game hops as follows.

- **Game 0:** This is the original IND-CPA game for our HDRS.

1. The simulator $\mathcal{S}$ runs $\mathsf{HDRS.Setup}$ to get system parameters $\mathsf{param}^{\mathsf{ch}}_{\mathsf{HDRS}} =< G_2, p, g, H_2 >$. Then, run $\mathsf{HDRS.KeyGen}$ to output (pk_R, sk_R). $\mathcal{S}$ relays $(\mathsf{param}^{\mathsf{ch}}_{\mathsf{HDRS}}, pk_R)$ to $\mathcal{A}$.
2. $\mathcal{A}$ randomly chooses two messages of the same length $m_0, m_1 \in \{0,1\}^*$ where $|m_0| = |m_1|$, and generates $sk_S = (x_0, x_1, x_2)$. $\mathcal{A}$ sends m_0, m_1 to $\mathcal{S}$. $\mathcal{S}$ flips a coin $coin \leftarrow \{0,1\}$ and generates $c_{coin} = \mathsf{HDRS.Signcrypt}(m_{coin}) = (c_{coin,0}, c_{coin,1}, c_{coin,2}, c_{coin,3})$ by:

$$\beta \xleftarrow{R} Z_p^*, \quad c_{coin,0} = g^{\beta}, \quad c_{coin,1} = y_{1,S}^{\beta},$$

$$c_{coin,2} = g^{m_{coin}}(h \cdot y_{0,R})^{\beta}, \quad c_{coin,3} = y_{1,R}^{m_{coin}} \cdot (y_{2,R}^{\beta})^{x_{1,S}}$$

$\mathcal{S}$ sends c_{coin} to $\mathcal{A}$.

3. Finally, $\mathcal{A}$ outputs a guess by $coin' \in \{0,1\}$. If $coin = coin'$, $\mathcal{A}$ wins and $\mathcal{B}$ outputs 1; else, $\mathcal{B}$ outputs 0.

- **Game 1:** The same as Game 0 except that $\mathcal{S}$ changes $y_{0,R}^{\beta}$ with $R_0 \in G_2$ during step 2 while computing c_{coin}:

$$h = H_2(\mathsf{CID}, pk_R), \quad \beta \xleftarrow{R} Z_p^*, \quad \boxed{R_0} \xleftarrow{R} G_2,$$

$$c_{coin,0} = g^{\beta}, \quad c_{coin,1} = y_{1,S}^{\beta}, \quad c_{coin,2} = g^{m_{coin}} h^{\beta} \boxed{R_0}, \quad c_{coin,3} = y_{1,R}^{m_{coin}} \cdot (y_{2,R}^{\beta})^{x_{1,S}}$$

- **Game 2:**: The same as Game 1 except that $\mathcal{S}$ changes $y_{2,R}^{\beta}$ with $R_1 \in G_2$ during step 2 while computing c_{coin}:

$$h = H_2(\mathsf{CID}, pk_R), \quad \beta \xleftarrow{R} Z_p^*, \quad R_0, \boxed{R_1} \xleftarrow{R} G_2,$$

$$c_{coin,0} = g^{\beta}, \quad c_{coin,1} = y_{1,S}^{\beta}, \quad c_{coin,2} = g^{m_{coin}} h^{\beta} R_0, \quad c_{coin,3} = y_{1,R}^{m_{coin}} \cdot \boxed{R_1}^{x_{1,S}}$$

Each next hop in above games only made negligible change to the former one, i.e., the modification of parameters are beyond adversary $\mathcal{A}$'s view; otherwise, we can construct an algorithm $\mathcal{B}$ to solve DDHP by using $\mathcal{A}$ who can distinguish between **Game 0** and **Game 1**, or **Game 1** and **Game 2**. Take **Game 0** and **Game 1** as an example:

On given a DDHP instance $g, g^a, g^b, R \in G$, $\mathcal{B}$ decides by proceeding the following game:

1. $\mathcal{B}$ chooses $x_1, x_2 \xleftarrow{R} Z_p$, it sets $y_0 = g^a$ and computes $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$. Next, $\mathcal{B}$ replays $pk_R = (y_0, y_1, y_2)$ and $\mathsf{param}_{\mathsf{HDRS}}$ to $\mathcal{A}$
2. $\mathcal{A}$ generates $sk_S = (x_0, x_1, x_2)$ as $\mathsf{HDRS.KeyGen}$. Then, it samples $m_0, m_1 \xleftarrow{R} \{0,1\}^*$ and relays them to $\mathcal{B}$. $\mathcal{B}$ flips a coin $coin \leftarrow \{0,1\}$ and generates $c_{coin} = (c_{coin,0}, c_{coin,1}, Pc_{coin,2}, c_{coin,3})$ as:

$$h = H_2(\mathsf{CID}, pk_R), \quad \beta \xleftarrow{R} Z_p^*,$$

$$c_{coin,0} = g^{\beta}, \quad c_{coin,1} = y_{1,S}^{\beta}, \quad c_{coin,2} = g^{m_b} h^{\beta} Z, \quad c_{coin,3} = y_{1,R}^{m_{coin}} \cdot (y_{2,R}^{\beta})^{x_{1,S}}$$

3. Finally, $\mathcal{A}$ outputs his guess $coin' \in \{0,1\}$. If $coin' = coin$, $\mathcal{A}$ wins and $\mathcal{B}$ outputs 1; else, $\mathcal{B}$ outputs 0.

Denote E_i as the event of **Game i** won by $\mathcal{A}$ (i.e. $1 \leftarrow \mathcal{S}$). If $Z = g^{ab}$ holds, this implies **Game 0**; else, if $Z \xleftarrow{R} G_2$, it implies **Game 1**. So, we can bound $\mathcal{B}$'s advantage in solving DDHP by $Adv_{\mathcal{B}}^{DDHP} = |Pr[E_0] - Pr[E_1]|$. Analogically, if $\mathcal{A}$ can distinguish between **Game 0** and **Game 1**, we can also bound $\mathcal{B}$'s advantage in solving DDHP by $Adv_{\mathcal{B}}^{DDHP} = |Pr[E_1] - Pr[E_2]|$.

In Game 2, $c_{coin} = \{c_{coin,0}, c_{coin,1}, c_{coin,2}, c_{coin,3}\}$ is generated follows one-time pad (concretely, $c_{coin,0}$ and $c_{coin,1}$ are set as $\beta \xleftarrow{R} Z_p^*$, $c_{coin,2}$ is set as $R_0 \xleftarrow{R} G_2$, $c_{coin,3}$ is set as $R_1 \xleftarrow{R} G_2$), and therefore, $\mathcal{A}$'s view is beyond the random coin $coin$, therefore, his advantage in winning Game 2 is negligible, i.e., $Pr[E_2] = \frac{1}{2}$. Analogically, from all above, we can bound the adversary $\mathcal{A}$'s advantage in breaking our IND-CPA security with $\mathcal{B}$ which solves the DDHP as: $Adv_{\mathcal{A}}^{IND-CPA} \leq 2 \cdot Adv_{\mathcal{B}}^{DDHP}$. Since $Adv_{\mathcal{B}}^{DDHP}$ is negligible, therefore, $Adv_{\mathcal{A}}^{IND-CPA}$ is negligible as well. Refer to [52] for more details.

Weak Unforgeability (WUF): We briefly show how to construct an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve CDHP [51]. On given a CDHP instance (p, g, g^A, g^B), $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

$\mathcal{B}$ derives $\mathsf{param}_{\mathsf{HDRS}}^{\mathsf{ch}} \leftarrow \mathsf{HDRS.Setup}$, and randomly chooses $x_{0,R}$ and $x_{2,R}$ from Z_p. It generates $y_{0,R} = g^{x_{0,R}}$ and $y_{2,R} = g^{x_{2,R}}$. Then, $\mathcal{B}$ sets $y_{1,R} = g^B$ and $y_{1,S} = g^A$. $\mathcal{B}$ relays above information to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $(m^*, \mathsf{CID}^*, c^*)$ where $c^* = (c_0^*, c_1^*, c_2^*, c_3^*)$ denotes forged signcryption under corresponding customized identity CID^* and message m^*.

$\mathcal{B}$ can compute g^{AB} as the solution to CDHP instance by (p, g, g^A, g^B) by:

$$g^{AB} = \left(\frac{c_3^*}{{c_1^*}^{x_{2,R}}}\right) \log_g {c_0^*}^{\frac{c_2^*}{{c_0^*}^{(e+x_{0,R})}}}{}^{-1}$$

Therefore, we can bound $\mathcal{B}$'s advantage by $Adv_{\mathcal{A}}^{WUF} \leq Adv_{\mathcal{B}}^{CDHP}$. Since CDHP is hard, $Adv_{\mathcal{B}}^{CDHP}$ is negligible and we can deduce that $Adv_{\mathcal{A}}^{WUF}$ is negligible as well. We omit details here.

Soundness of Homomorphic Integration: Suppose $\mathcal{A}$ is an efficient adversary who breaks the soundness of homomorphic integration of our HDRS, we can construct an algorithm $\mathcal{B}$ to solve q-SDHP [50] as below:

On given a q-SDHP instance $(A_0, \cdots, A_q)$ where A_i is denoted as $A_i = g^{x_i} \in G_2$ for $i = 1, \cdots, q$ and $A_0 = g$. $\mathcal{B}$ interacts with $\mathcal{A}$ as below:

$\mathcal{B}$ picks a security parameter λ and runs $\mathsf{param}_{\mathsf{HDRS}}^{\mathsf{ch}} \leftarrow \mathsf{HDRS.Setup}(\lambda)$. Then, sample (sk_R, pk_R) and (sk_S, pk_S). Randomly pick customized identity CID and message $m \in \{0,1\}^*$, and run $c \leftarrow \mathsf{HDRS.Signcrypt}(\mathsf{CID}, \mathsf{m}, \mathsf{sk_S}, \mathsf{pk_R})$ to derive ciphertext c. Next, run $chal \leftarrow \mathsf{HDRS.IntChal}()$ with no input to derive a challenge $chal$ for homomorphic integration. To note, we set n target messages to be integrated by 1 for ease of analysis. Then, no homomorphic operation is performed but a proof is generated. This will not affect our theoretical analysis but will save much space to specify redundant parameters. Finally, $\mathcal{B}$ sends the above generated information to $\mathcal{A}$ and $\mathcal{A}$ is challenged to produce a proof $proof^*$ such that $1 \leftarrow \mathsf{HDRS.IntVerify}(\mathsf{chal}, \mathsf{proof}^*, \mathsf{pk_R})$.

Specifically, denote $proof$ and $proof^*$ as the valid and forged proof for c (target ciphertexts to be integrated) on a challenge c, respectively. If $proof = proof^*$, we can reduce it to the public collision-resistance of our HCCH by viewing $y_{0,R}$ as hash key hk and $x_{0,R}$ as trapdoor key tk for HCCH scheme, as we discussed in 4.2 earlier. Shortly, an algorithm (say $\mathcal{C}$) can be constructed to solve q-SDHP with non-negligible advantage using the adversary $\mathcal{A}$ who breaks the soundness of homomorphic integration of our HDRS. For $proof \neq proof^*$, we have following equations where $e = H_2(\mathsf{CID}, pk_R)$ and CID denotes a customized identity:

$$\begin{aligned} g^{m'}(g^{\alpha'})^{e+x_{0,R}} &= g^m(h \cdot y_{0,R})^\alpha, \\ g^{m^*}(g^{\alpha^*})^{e+x_{0,R}} &= g^m(h \cdot y_{0,R})^\alpha. \end{aligned}$$

By considering these equations, we can also build algorithm, say $\mathcal{C}$, to solve q-SDHP with non-negligible advantage. Specifically, this implies the breaking of a stronger security notion called "uniqueness" (as identified in [22], which asks that it is infeasible to come up with two randomness to hold collision for the same challenged message). Due to space limitation, details are omitted.

Based on the above, we can reduce the soundness of homomorphic integration of our HDRS to the public collision-resistance of our HCCH as proven in section 4.2 earlier.

Non-repudiation: Suppose $\mathcal{A}$ is an efficient adversary who breaks the non-repudiation of our HDRS, we briefly show how to construct an algorithm $\mathcal{B}$ to solve q-SDHP by using adversary $\mathcal{A}$ as follows. Given a q-SDHP instance $(g, g^x, \cdots, g^{x^q})$, in order to derive an $(c, g^{\frac{1}{(x+c)}})$ for some $c \in Z_q^*$, $\mathcal{B}$ proceeds with $\mathcal{A}$ as follows:

$\mathcal{B}$ picks a security parameter λ and runs $\mathsf{param}^{\mathsf{ch}}_{\mathsf{HDRS}} \leftarrow \mathsf{HDRS.Setup}(\lambda)$. Then, sample (sk_R, pk_R) and (sk_S, pk_S). Randomly pick customized identity CID and message $m \in \{0,1\}^*$, and run $c \leftarrow \mathsf{HDRS.Signcrypt}(\mathsf{CID}, \mathsf{m}, \mathsf{sk_S}, \mathsf{pk_R})$ to derive ciphertext c. Next, run $chal \leftarrow \mathsf{HDRS.IntChal}()$ with no input to derive a challenge $chal$ for homomorphic integration. To note, we set n target messages to be integrated by 1 for ease of analysis. So, no homomorphic operation is performed but a proof is generated still. This will not affect our theoretical analysis but will save much space to specify redundant parameters. Next, $\mathcal{B}$ sends above generated information to $\mathcal{A}$ and $\mathcal{A}$ is challenged to produce a proof $proof^*$. On given the $proof^*$ by $\mathcal{A}$, $\mathcal{B}$ runs (0 or 1) $\leftarrow \mathsf{HDRS.IntVerify}(\mathsf{chal}, \mathsf{proof}^*, \mathsf{pk_R})$ to check validity. If the output is 0, $\mathcal{A}$ outputs $\perp$; if the output is 1, $\mathcal{B}$ can extract an answer to q-SDHP from $proof^*$ by viewing Z_2 as $g^{\alpha *}$, Z_3 as $g^{\alpha **}$, $y_{0,R}$ as hash key hk, $x_{0,R}$ as the trapdoor key tk, $H_1(M^*)$ and $H_1(M^{**})$ as m and m' in our proposed HCCH. Then, $\mathcal{B}$ can compute $\pi = (s^* \cdot \sum_{i=1}^{q_s}(A_i)^{-\gamma_i})^{\frac{1}{\gamma-1}} = g^{\frac{1}{x+e^*}}$ as an answer to q-SDHP the same way as discussed in section 4.2 for HCCH. Here, $e^* = g^{H_2(\mathsf{CID}, pk_R)}$ and $s^* = (\frac{g^{\alpha^{**}}}{g^{\alpha^*}})^{\frac{1}{H_1(M^* - H_1(M^{**}))}}$. Therefore, we successfully reduce the security of non-repudiation of our HDRS to the public collision-resistance of our HCCH.

6 INSTANTIATIONS

In this section, we give instantiations of our HCCH and HDRS schemes for practical use. We also briefly instantiate technical details of a blockchain with proof-of-discrete-logarithm to improve the performance of our HDRS by shifting computations of the discrete logarithm to miners who maintain the blockchain.

6.1 Instantiating HCCH

We show how to use HCCH to process mass data to be outsourced where it maps any arbitrary size of data (i.e., M) to short and fixed-length hash value ($\mathcal{H} \leftarrow \mathsf{HCCH.Hash(M)}$).

The result will be recorded on the blockchain as a proof of outsourcing, and the user can be rewarded with some crypto-currencies as incentives.

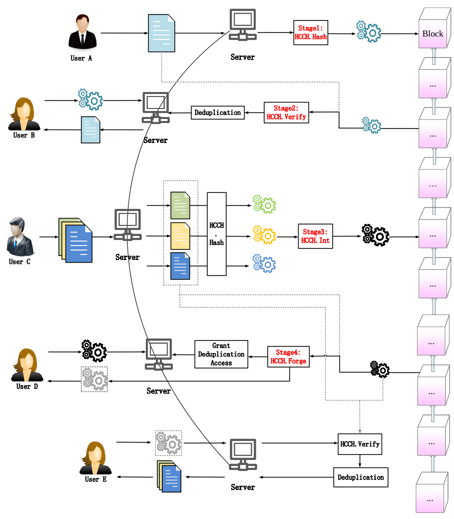


Fig. 2. Instantiation of HCCH

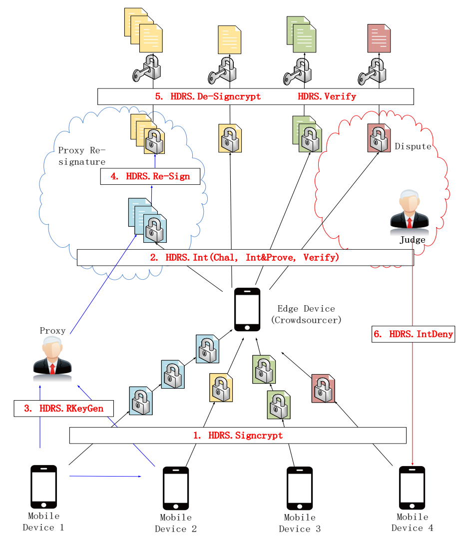


Fig. 3. Instantiation of HDRS

As it is shown in Figure 2, when user A outsources a file to the server, the hash of the file (generated by HCCH.Hash at stage 1, marked in red) is recorded in blockchain as an evidence. Later, if user B wishes to access this file, she only needs to show the hash proof to the corresponding server. Then, a verification (by HCCH.Verify at stage 2, marked in red) is executed by comparing with the record on the blockchain. If the verification passes, access to the file can be granted accordingly. This process is known as deduplication [18], where the outsourcing of repeated files will be deleted to save both bandwidth and storage.

In addition, the hash values of multiple outsourced files (say, three files uploaded by user C) can be integrated (by HCCH.Int at stage 3, marked in red) to one value. This can be used as a proof of ownership for user C, and it will be recorded on the blockchain as well. Accordingly, if user D also shares these files with user C , to differentiate, the HCCH.Forge algorithm (at stage 4, marked in red) can be executed to produce a hash collision where the hash value remains the same but new chameleon hash randomness is used to certify D's ownership meanwhile differentiating it with C's. This allows deduplication to take effect without causing any contradictions [53]. Meanwhile, the access control mechanism can be created for C and D by identifying the signed chameleon hash randomness. Analogically, user E can also retrieve these files which he co-owned with user C and D.

A concrete construction of the above scenario is presented in our other work, consequently, deduplication, data auditing and dynamic update are achieved in one framework by the use of blockchain.

6.2 Instantiating HDRS

We instantiate how to use our HDRS as the basic privacy protection to outsource data at small scale to an edge device. To capture the business model, inspired by Yang et al [54], we instantiate it by crowdsourcing-based crowdsensing (CbC). In this scenario, our HDRS is applied to outsourced data, and the user can gain profits from outsourcing them to a crowdsourcer (the price of an outsourced file is negotiated between user and receiver (i.e., the crowdsourcer)). How to put price on each data is out of the scope of this paper, but we will discuss this interesting topic in our next work under the background of datamarket.

In Figure 3, an edge device which also known as crowdsourcer employs several other mobile devices to collect sensing data. The transmission of these sensitive data is supposed to be encrypted and signed, since they are transmitted on public channel (where tampering and eavesdropping may happen). Concretely, mobile device (say, 1,2,3,4) run HDRS.Signcrypt (at stage 1) to signcrypt sensing data with fixed-length (we can partition file with arbitrary length into chunks which fixed size by hash function, or specifically, via our HCCH). On receiving the information, crowdsourcer can perform homomorphic integration (at stage 2) on the received data, this avoids repeated decryption (in stage 4) because sensing data (like signal) often comes at the consecutive manner and decrypting them one by one is inefficient. Additionally, the mobile device can designate a proxy (at stage 3) to re-sign (at stage 4) a signcrypted information (previously generated by him) to another user (say, from device 1 to device 2) by the notion of proxy re-signature [55]. This allows users to trade their sensing data with others and gain profits. Last but not least, if there exists any disputes on the received information, a trusted judge can be entrusted to contact the corresponding user to check the validity (via HDRS.IntDeny, at stage 6). As undeniable authentication (as guarded by our non-repudiation security in section 5.2) is achieved, it can be used as an evidence for information forensics [56] purposes.

Algorithm 1 Block Generation

Input: On input n transactions $tran = \{tran_i\}_{1 \leq i \leq n}$ broadcast in a current network, where each $tran_i$ is embedded with a signed discrete logarithm $SIGN_{sk_i}(g^{m_i})$ as a puzzle to be solved. It is issued and signed by a user with signing key sk_i where we instantiate SIGN by BLS signature [49].

Output: New block $\mathcal{BO}$ or $\perp$

1: Verify the validity of each $tran_i$ by user ith's public key pk_i by [49]. If invalid, output $\perp$; else continue.
2: Define ComptDL as an algorithm to calculate discrete logarithm (e.g., [57, 58]), run $m_i \leftarrow \mathsf{ComptDL}(g^{m_i})$ for each $1 \leq i \leq n$.
3: Aggregate $m = m_1 \parallel \cdots \parallel m_n$ and sign it with miner's signing key (say, sk_π) such that $\sigma = SIGN_{sk_\pi}(m)$. Here, miner is the one who succeeded in performing above steps.
4: Include $\{(m, \sigma), pk_\pi\}$ in the block header where pk_π is the public key of miner, and include $tran$ in block body for public auditing. Apply other definitions as described in [11].

6.3 Instantiating B-PoDL

We briefly instantiate how to construct a new blockchain with proof-of-discrete-logarithm (B-PoDL) to help solve discrete logarithm (caused by HDRS.De-Signcrypt) via miners. The trick is to embed an discrete logarithm instance for each transaction. Our proposal is adapted from a simplified version of bitcoin [11], instantiation of block generation is given in algorithm 1. For auditing a block, block verification is instantiated in algorithm 2.

As noted previously, we only focus on the core design of blockchain. We refer readers to [59, 60] for questions related to incentives.

Algorithm 2 Block Verification

Input: On input a block $\mathcal{BO}$ which incorporates $\{m', \sigma', tran', pk_\pi\}$ where $tran'$ denotes instances of discrete logarithms and m' denotes an answer for it.

Output: 0 or 1

Verify the validity of σ' by miner's private key pk_π. If invalid, output 0; else continue.

Verify the validity of signature for each $tran_i$ included in $tran$ by public key pk_i of the ith user. If invalid, output 0; else continue.

Check whether $g^{m_i'} = g^{m_i}$ holds for each $1 \leq i \leq n$. If all hold, output 1; else, output 0.

Table 3. Definitions of Primitive Operations

Symbol	Meaning	Approximation	Symbol	Meaning	Approximation
T_m	Multiplication		T_e	Exponentiation	$\approx 21T_m$
T_i	Inversion	$\approx 11.6T_m$	T_p	Pairing	$\approx 87T_m$
T_{pa}	Point Addition	$\approx 0.12T_m$	T_{pm}	Point Multiplication	$\approx 22T_m$
T_{mpm}	Multi-Point Multiplication	$\approx 29T_m$	T_{mtp}	Map-to-Point Function	$\approx 29T_m$
T_{sig}	Signing of BLS [49]	$\approx 58T_m$	T_{mtp}	Verifying of BLS [49]	$\approx 203T_m$
T_{mht}	Constructing Merkle-Hash Tree		T_{log}	Computing Discrete Logarithm	$\geq \Omega(p^{\frac{1}{2}})$ [63]

We use T_m as unit of measurement and derive approximations based on indications of [39, 64]. According to [63], the lower bound for general discrete logarithm is $\Omega(p^{\frac{1}{2}})$ where p denotes the largest prime factor of group order and no special property is exploited in this group.

7 PERFORMANCE EVALUATION

In this section, we give a comprehensive evaluation of our proposed HCCH, HDRS and B-PoDL. It includes both complexity and experiment analysis.

7.1 Complexity Analysis

To start, we first give some definitions for involved cryptographic operations in Table 3. For ease of evaluation, we use group multiplication (T_m) as a unit of measurement, and convert primitive operation by the complexity of T_m based on [61, 62] where it is possible.

According to Table 4, our proposed HCCH is less efficient than relevant schemes (e.g., 67% and 58% slower than scheme [25] in hashing and forging respectively). Meanwhile, according to Table 5 and Figure 7(b), our proposed HDRS ranks the 6th efficient one in signcryption among 10 listed works in comparison. Assuming $|G| = |G_1| = |G_2| = |G_T| = |Z_q^*|$, our HDRS costs the least communication overhead. Therefore, our proposals are acceptably efficient from complexity perspective.

7.2 Experiment Analysis

To confirm our theoretical analysis, we utilize dev3 (as specified in Table 6) to simulate our proposed hashing scheme HCCH. Configuration is given in Table 7. For ease of comparison, we use SHA-256 as benchmark scheme. According to Figure 5, our HCCH is less efficient than SHA-256 in terms of hashing and verification. This coincides with our complexity analysis. However, since the performance of HCCH is dominated by throughput efficiency of

Table 4. Complexity of Hash Schemes

Scheme	Hash	Forge
HCCH	$5T_e + 2T_m \approx 107T_m$	$2T_e + 4T_m + 2T_i \approx 69.6T_m$
BRCB[65]	$3T_e + T_m \approx 64T_m$	$(k+1)T_e + 3T_m + T_i \geq 56.6T_m$
ICHA[20]	$4T_e + 2T_m \approx 86T_m$	$2T_e + 2T_m + T_i \approx 57.6T_m$
CHKE[24]	$3T_e + 2T_m \approx 65T_m$	$2T_e + 2T_m + T_i \approx 57.6T_m$
KEFC[25]	$3T_e + 1T_m \approx 64T_m$	$2T_e + 2T_m \approx 44T_m$

We set $k = 1$ to derive the a lower bound for the forging algorithm of [65] where k denotes number of threshold parties to forge a collision.

Table 5. Complexity of Signcryption Schemes

Scheme	Signcryption	De-Signcryption	Communication
PSIB[39]	$4T_e \approx 84T_m$	$2T_e + 2T_p + 2T_i \approx 227.6T_m$	$2\|G_1\| + 2\|G_2\| + \|Z_q^*\|$
AISI[35]	$4T_e \approx 84T_m$	$6T_p + T_i \approx 533.6T_m$	$4\|G_1\| + \|G_2\|$
IOSL[31]	$2T_{pm}+T_{mpm}+T_e+2T_m \approx 96T_m$	$3T_{pm} + T_e + 2T_p \approx 261T_m$	$3\|G\| + n + 2\|Z_q^*\|$
EIOE[34]	$T_{mpm} + 3T_{pm} + 2T_m \approx 97T_m$	$T_p + T_{pm} + 2T_m + 2T_i \approx 122.6T_m$	$2\|G\| + n + 2\|Z_q^*\|$
OOIB[32]	$4T_{pm} + T_{mpm} + 3T_m \approx 120T_m$	$T_{mpm} + T_p \approx 116T_m$	$4\|G\| + n + 3\|Z_p\|$
HDRS	$8T_e + 3T_m \approx 171T_m$	$T_e+T_m+T_i+T_{log} \geq 33.6T_m+T_{log}$	$4\|G\|$
FSIS[37]	$4T_e + T_p \approx 171T_m$	$6T_p + T_i \approx 533.6T_m$	$4\|G_1\| + 2\|G_2\|$
IBSS[38]	$6T_e + T_p \approx 213T_m$	$2T_e + 6T_p + T_i \approx 575.6T_m$	$4\|G_1\| + \|Z_q^*\|$
SISS[36]	$6T_e + T_p \approx 213T_m$	$2T_e + 6T_p + T_i \approx 575.6T_m$	$4\|G_1\| + \|G_2\| + \|Z_q^*\|$
IBOO[33]	$6T_{mpm} + 2T_e + 2T_m \approx 218T_m$	$2T_p + 5T_{mpm} \approx 319T_m$	$\|G_T\| + 5\|G_T\| + n + 2\|Z_p^*\|$

Listed by signcryption cost from low to high. Each scheme is named by the abbreviation of title.

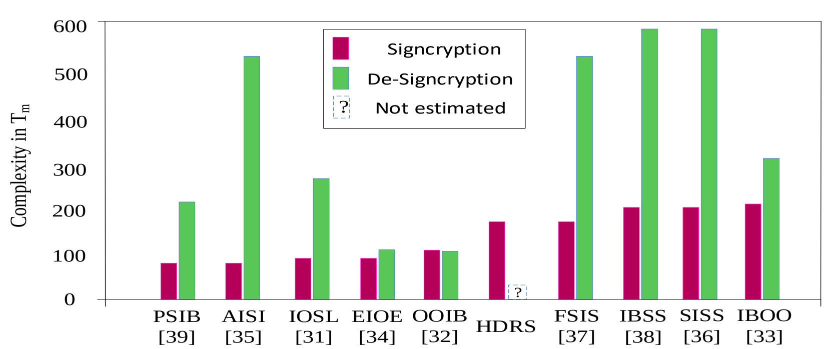


Fig. 4. Overview of Signcryption Schemes

dev3 (other than inherent cryptographic operations), our HCCH suffices to be implemented for practical use.

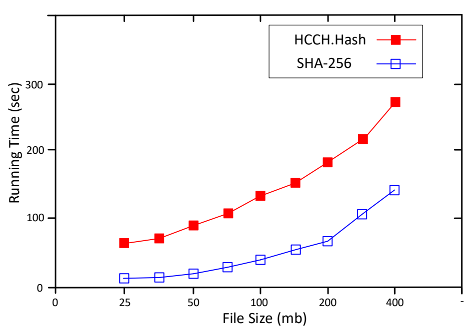
Next, we utilize dev1 (as specified in Table 6) to test performance of our HDRS. We use RSA as benchmark scheme as it is widely used. Configuration is given in Figure 6(a). According to Figure 6(b), both tested algorithms show poor performance in processing massive files. Meanwhile, according to Figure 7(b), our HDRS scales poor in decryption due to intractability of solving discrete logarithm. Here, we try to simulate decryption cost by extracting results directly from [58]. In a previous research [58], a unique group with

Table 6. Simulation Devices

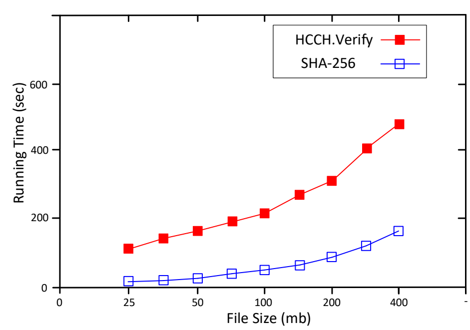
Parameter	Dev1	Dev2	Dev3
Model	Dell Inspiron 15 7567	Alienware 15 R3	Raspberry Pi 3 Model B
CPU	4×(3.5GHz)	4×(2.8GHz)	1.2 GHz
RAM	8 GB	16 GB	1 GB
Storage	256 GB SSD	1T+512GB HDD+SSD	32 GB Micro SD
GPU	GTX 1050	GTX 1070	N/A

Table 7. Configuration of HCCH

Platform OS	Dev3 Windows 10
Library	JPBC [66]
Curve	$y^2 = x^3 + 1$
Benchmark	SHA-256 [67]
Simulating HCCH.Hash	$MapToGroup$ instantiated by [49]: $\{0,1\}^* \to G^*$
Simulating HCCH.Verify	DDH Oracle based on [49, 68]



(a) Performance of HCCH.Hash

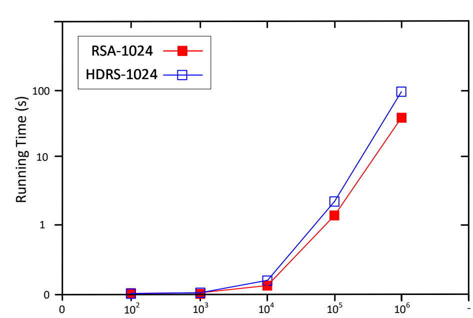


(b) Performance of HCCH.Verify

Fig. 5. Performance of HCCH

Platform OS&Compiler	Dev1 Win10 & C
Bechmark	RSA-1024 [69]
Key Length	1024 bits
Message Size (per)	117 bytes
Tested Trials	100
Size of Tested File (approx.)	From 11.4 kb to 111.57 mb

(a) Configuration of RSA and HDRS



(b) Comparison of Encryption Cost

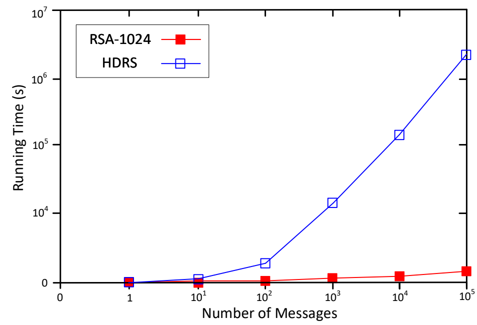
Fig. 6. Performance of HDRS in Signcryption

smoothness-order [70] and graphics processing unit (GPU) are exploited to solve small discrete logarithm. According to results, decryption costs are high.

Inspired by blockchain [11], we try to shift decryption burdens from user-side to distributed devices (i.e., miners). We propose a notion called blockchain with proof-of-discrete-logarithm (B-PoDL) which is similar to hashing-based proof-of-work consensus mechanism. Differently,

	Est. T_{log} by [58]	
Modulus Length (bit)	Group Smoothness	Time Cost (s)
1536	2^{48}	≈ 23
1536	2^{50}	≈ 32
	$\cdots$	
1536	2^{58}	≈ 270
Other Parameter	Platform: Dev2 The rest as Fig.6 (a)	

(a) Configuration of Decryption



(b) Performance of Decryption Cost

Fig. 7. Comparison of Decryption

it works by performing repeated computation on small discrete logarithm as discussed in [58]. Performance is verified as follows. We conduct experiment through Geth 1.8.23 by dev2 (specified in Table 6). Due to limit on mining power, we set mining difficulty to low level. We randomly sampled $25,000$ transactions, and use dev1 to transmit them to dev2 via OpenSSL. Based on the above, we set checkpoint for transaction number at $1,000$, $5,000$ and $25,000$ for analysis. Computing costs on small discrete logarithm are drawn from [58] (as squared in Figure 7, 23 sec for decryption inp a unique group). The results reported in Table 8 demonstrate the cost to append a block in a basic chain and B-PoDL. As it is shown in Table 8, our B-PoDL scales fine with increasing number of transactions. Therefore, our B-PoDL is helpful to shift burdens of computing discrete logarithm from the user device meanwhile issuing financial incentives to users (e.g., issuing bitcoin) to prosper outsourcing business. This further indicates the possibility of applying our HDRS scheme for practical use, as well as realizing HUCDO model we introduced.

Table 8. Overview of Chain Growth

Number of Transactions	Running Time (min) basic chain/B-PoDL	Confirmation Rate basic chain/B-PoDL	Chain Growth (mb)
$1,000$	3/8.21	33/121 per min	≈ 1.2
$5,000$	53/75.24	94/66 per min	≈ 32
$25,000$	250.18/325.59	100/74 per min	≈ 172

8 CONCLUSION

In this paper, we proposed a hybrid user-centric data outsourcing scheme which considers user's benefits in security, efficiency and economy. Our contributions are three-fold: theories, implementations and evaluations. Firstly, we proposed HCCH and HDRS as two fundamental tools to enable hybrid outsourcing. Secondly, we instantiate how to use HCCH and HDRS for practical implementations. Additionally, a new blockchain called B-PoDL is instantiated to improve the performance of our proposal meanwhile serving as an incentive method to encourage users for outsourcing. Finally, evaluations showed that our proposals are efficient, secure and economic for users to outsource data from cyber-physical systems to the cloud server and edge devices.

REFERENCES

[1] Edward A Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE, 2008.

[2] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23, 2015.

[3] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[4] Deepak Puthal, Rajiv Ranjan, Surya Nepal, and Jinjun Chen. Iot and big data: An architecture with data flow and security issues. In *Cloud infrastructures, services, and IoT systems for smart cities*, pages 243–252. Springer, 2017.

[5] Anthony D JoSEP, RAnDy KAtz, AnDy KonWinSKi, LEE Gunho, DAViD PAttERSon, and ARiEL RABKin. A view of cloud computing. *Communications of the ACM*, 53(4), 2010.

[6] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *International Conference on Financial Cryptography and Data Security*, pages 477–498. Springer, 2016.

[7] Deepak Puthal, Saraju P Mohanty, Priyadarsi Nanda, and Uma Choppali. Building security perimeters to protect network systems against cyber threats [future directions]. *IEEE Consumer Electronics Magazine*, 6(4):24–27, 2017.

[8] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE, 1997.

[9] Deepak Puthal, Rajiv Ranjan, Ashish Nanda, Priyadarsi Nanda, Prem Prakash Jayaraman, and Albert Y Zomaya. Secure authentication and load balancing of distributed edge datacenters. *Journal of Parallel and Distributed Computing*, 124:60–69, 2019.

[10] Jian-Jun Wang and De-Li Yang. Using a hybrid multi-criteria decision aid method for information systems outsourcing. *Computers & Operations Research*, 34(12):3691–3700, 2007.

[11] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[12] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609. Acm, 2007.

[13] Hugo Mario Krawczyk and Tal D Rabin. Chameleon hashing and signatures, August 22 2000. US Patent 6,108,783.

[14] Yuliang Zheng. Digital signcryption or how to achieve cost (signature & encryption)? cost (signature)+ cost (encryption). In *Annual International Cryptology Conference*, pages 165–179. Springer, 1997.

[15] Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597. Acm, 2007.

[16] Giuseppe Ateniese, Roberto Di Pietro, Luigi V Mancini, and Gene Tsudik. Scalable and efficient provable data possession. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, page 9. ACM, 2008.

[17] Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou. Enabling public verifiability and data dynamics for storage security in cloud computing. In *European symposium on research in computer security*, pages 355–370. Springer, 2009.

[18] John R Douceur, Atul Adya, William J Bolosky, P Simon, and Marvin Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *Proceedings 22nd international conference on distributed computing systems*, pages 617–624. IEEE, 2002.

[19] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 296–312. Springer, 2013.

[20] Giuseppe Ateniese and Breno de Medeiros. Identity-based chameleon hash and applications. In *International Conference on Financial Cryptography*, pages 164–180. Springer, 2004.

[21] Giuseppe Ateniese and Breno de Medeiros. On the key exposure problem in chameleon hashes. In *International Conference on Security in Communication Networks*, pages 165–179. Springer, 2004.

[22] Jan Camenisch, David Derler, Stephan Krenn, Henrich C Pöhls, Kai Samelin, and Daniel Slamanig. Chameleon-hashes with ephemeral trapdoors. In *IACR International Workshop on Public Key Cryptography*, pages 152–182. Springer, 2017.

[23] Stephan Krenn, Henrich C Pöhls, Kai Samelin, and Daniel Slamanig. Chameleon-hashes with dual long-term trapdoors and their applications. In *International Conference on Cryptology in Africa*, pages 11–32. Springer, 2018.

[24] Xiaofeng Chen, Fangguo Zhang, and Kwangjo Kim. Chameleon hashing without key exposure. In *International Conference on Information Security*, pages 87–98. Springer, 2004.

[25] Xiaofeng Chen, Fangguo Zhang, Haibo Tian, Baodian Wei, and Kwangjo Kim. Key-exposure free chameleon hashing and signatures based on discrete logarithm systems. *IACR Cryptology ePrint Archive*, 2009:35, 2009.

[26] Xiaofeng Chen, Haibo Tian, Fangguo Zhang, and Yong Ding. Comments and improvements on key-exposure free chameleon hashing based on factoring. In *International Conference on Information Security and Cryptology*, pages 415–426. Springer, 2010.

[27] Mihir Bellare and Todor Ristov. A characterization of chameleon hash functions and new, efficient designs. *Journal of cryptology*, 27(4):799–823, 2014.

[28] Giuseppe Ateniese, Daniel H Chou, Breno De Medeiros, and Gene Tsudik. Sanitizable signatures. In *European Symposium on Research in Computer Security*, pages 159–177. Springer, 2005.

[29] Marc Fischlin and Patrick Harasser. Invisible sanitizable signatures and public-key encryption are equivalent. In *International Conference on Applied Cryptography and Network Security*, pages 202–220. Springer, 2018.

[30] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. In *International Workshop on Practice Theory in Public Key Cryptosystems: Public Key Cryptography*, 2002.

[31] Fagen Li, Muhammad Khurram Khan, Khaled Alghathbar, and Tsuyoshi Takagi. Identity-based online/offline signcryption for low power devices. *Journal of Network Computer Applications*, 35(1):340–347, 2012.

[32] Joseph K. Liu, Joonsang Baek, and Jianying Zhou. *Online/Offline Identity-Based Signcryption Revisited.* 2010.

[33] S. Sharmila Deva Selvi, S. Sree Vivek, and C. Pandu Rangan. Identity based online/offline encryption and signcryption schemes revisited. In *International Conference on Security Aspects in Information Technology*, 2011.

[34] Jianchang Lai, Mu Yi, and Fuchun Guo. *Efficient identity-based online/offline encryption and signcryption with short ciphertext.* 2017.

[35] Zhengping Jin, Qiaoyan Wen, and Hongzhen Du. An improved semantically-secure identity-based signcryption scheme in the standard model. *Computers Electrical Engineering*, 36(3):545–552, 2010.

[36] Fagen Li and Tsuyoshi Takagi. Secure identity-based signcryption in the standard model. *Mathematical Computer Modelling*, 57(11-12):2685–2694, 2013.

[37] Xiangxue Li, Haifeng Qian, Weng Jian, and Yu Yu. Fully secure identity-based signcryption scheme with shorter signcryptext in the standard model. *Mathematical Computer Modelling*, 57(3-4):503–511, 2013.

[38] S. Sharmila Deva Selvi, S. Sree Vivek, Dhinakaran Vinayagamurthy, and C. Pandu Rangan. Id based signcryption scheme in standard model. In *International Conference on Provable Security*, 2012.

[39] Arijit Karati, Sk Hafizul Islam, G. P. Biswas, Md Zakirul Alam Bhuiyan, Pandi Vijayakumar, Marimuthu Karuppiah, Arijit Karati, Sk Hafizul Islam, G. P. Biswas, and Md Zakirul Alam Bhuiyan. Provably secure identity-based signcryption scheme for crowdsourced industrial internet of things environments. *IEEE Internet of Things Journal*, PP(99):1–1, 2017.

[40] Feng Bao and Robert H. Deng. A signcryption scheme with signature directly verifiable by public key. *Proc Pck98 Feb*, 1431:55–59, 1998.

[41] Chandana Gamage, Jussipekka Leiwo, and Yuliang Zheng. Encrypted message authentication by firewalls. In *International Workshop on Public Key Cryptography*, 1999.

[42] John Malonelee and Wenbo Mao. Two birds one stone: Signcryption using rsa. In *Rsa Conference on the Cryptographers Track*, 2003.

[43] Liqun Chen and John Malone-Lee. Improved identity-based signcryption. *Lecture Notes in Computer Science*, 2004(1):362–379, 2005.

[44] Xavier Boyen. *Multipurpose Identity-Based Signcryption.* 2003.

[45] Sherman S. M. Chow, S. M. Yiu, Lucas C. K. Hui, and K. P. Chow. Efficient forward and provably secure id-based signcryption scheme with public verifiability and public ciphertext authenticity. 2003.

[46] Beno?t Libert and Jean Jacques Quisquater. A new identity-based signcryption scheme from pairings. *IEEE Information Theory Workshop*, pages 155–158, 2003.

[47] Tsz Hon Yuen and Victor K. Wei. Fast and proven secure blind identity-based signcryption from pairings. In *International Conference on Topics in Cryptology*, 2005.

[48] Paulo S. L. M. Barreto, Beno?t Libert, Noel Mccullagh, and Jean Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *International Conference on Theory Application of Cryptology Information Security*, 2005.

[49] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer, 2001.

[50] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 56–73. Springer, 2004.

[51] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[52] Fatemeh Rezaeibagha, Mu Yi, Shiwei Zhang, and Xiaofen Wang. Provably secure homomorphic signcryption. In *International Conference on Provable Security*, 2017.

[53] Ke Huang, Xiaosong Zhang, Xiaofen Wang, Xiaojiang Du, and Ruonan Zhang. Ebd-mle: Enabling block dynamics under bl-mle for ubiquitous data. In *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, pages 1281–1288. IEEE, 2017.

[54] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones. *IEEE/ACM Transactions on Networking (TON)*, 24(3):1732–1744, 2016.

[55] Giuseppe Ateniese and Susan Hohenberger. Proxy re-signatures: new definitions, algorithms, and applications. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 310–319. ACM, 2005.

[56] Matthew C Stamm, Min Wu, and KJ Ray Liu. Information forensics: An overview of the first decade. *IEEE access*, 1:167–200, 2013.

[57] Daniel J Bernstein and Tanja Lange. Computing small discrete logarithms faster. In *International Conference on Cryptology in India*, pages 317–338. Springer, 2012.

[58] Ryan Henry and Ian Goldberg. Solving discrete logarithms in smooth-order groups with cuda. In *Workshop Record of SHARCS*, pages 101–118, 2012.

[59] Yongjun Ren, Yepeng Liu, Sai Ji, Arun Kumar Sangaiah, and Jin Wang. Incentive mechanism of data storage based on blockchain for wireless sensor networks. *Mobile Information Systems*, 2018, 2018.

[60] Fan Zhang, Ittay Eyal, Robert Escriva, Ari Juels, and Robbert Van Renesse. {REM}: Resource-efficient mining for blockchains. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 1427–1444, 2017.

[61] Arijit Karati, SK Hafizul Islam, GP Biswas, Md Zakirul Alam Bhuiyan, Pandi Vijayakumar, and Marimuthu Karuppiah. Provably secure identity-based signcryption scheme for crowdsourced industrial internet of things environments. *IEEE Internet of Things Journal*, 5(4):2904–2914, 2018.

[62] Jonathan Katz, Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[63] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 256–266. Springer, 1997.

[64] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In *International Workshop on Public Key Cryptography*, pages 277–290. Springer, 2004.

[65] Ke Huang, Xiaosong Zhang, Yi Mu, Xiaofen Wang, Guomin Yang, Xiaojiang Du, Qi Xia, Fatemeh Rezaeibagha, and Mohsen Guizani. Building redactable consortium blockchain for industrial internet-of-things. *IEEE Transactions on Industrial Informatics*, 2019.

[66] Angelo De Caro and Vincenzo Iovino. jpbc: Java pairing based cryptography. In *2011 IEEE symposium on computers and communications (ISCC)*, pages 850–855. IEEE, 2011.

[67] Wikipedia contributors. Sha-2 — Wikipedia, the free encyclopedia, 2019. [Online; accessed 5-May-2019].

[68] Antoine Joux and Kim Nguyen. Separating decision diffie–hellman from computational diffie–hellman in cryptographic groups. *Journal of cryptology*, 16(4):239–247, 2003.

[69] Jiezhao Peng and Qi Wu. Research and implementation of rsa algorithm in java. In *2008 International Conference on Management of e-Commerce and e-Government*, pages 359–363. IEEE, 2008.

[70] Carl Pomerance and Igor E Shparlinski. Smooth orders and cryptographic applications. In *International Algorithmic Number Theory Symposium*, pages 338–348. Springer, 2002.