LRCoin: Leakage-Resilient Cryptocurrency Based on Bitcoin for Data Trading in IoT

Yong Yu, Yujie Ding, Yanqi Zhao, Yannan Li, Yi Zhao, Xiaojiang Du, *Senior Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—Currently, the number of Internet of Things (IoT) devices making up the IoT is more than 11 billion and this number has been continuously increasing. The prevalence of these devices leads to an emerging IoT business model called Device-as-a-service, which enables sensor devices to collect data disseminated to all interested devices. The devices sharing data with other devices could receive some financial reward, such as Bitcoin. However, side-channel attacks, which aim to exploit some information leaked from the IoT devices during data trade execution, are possible since most of the IoT devices are vulnerable to be hacked or compromised. Thus, it is challenging to securely realize data trading in IoT environment due to the information leakage, such as leaking the private key for signing a Bitcoin transaction in Bitcoin system. In this paper, we propose LRCoin, a kind of leakage-resilient cryptocurrency based on bitcoin in which the signature algorithm used for authenticating bitcoin transactions is leakage-resilient. LRCoin is suitable for the scenarios where information leakage is inevitable, such as IoT applications. Our core contribution is proposing an efficient bilinear-based continual-leakage-resilient ECDSA signature. We prove the proposed signature algorithm is unforgeable against adaptively chosen messages attack in the generic bilinear group model under the continual leakage setting. Both the theoretical analysis and the implementation demonstrate the practicability of the proposed scheme.**

***Index Terms*—Blockchain, data trading, generic bilinear group model, leakage resilient signature.**

I. Introduction

THE INTERNET of Things (IoT) has emerged as an area of incredible potential and impact. According to Gartner Inc. [1], more than 11 billion IoT devices have been connected to the IoT network in 2018. The number of these devices is continuously increasing and is expected to be 20 billion by 2020. The application of IoT pervades everywhere from smart home, smart cities, manufacturing, commerce, education to supply chain, logistics, and almost anything we can imagine [2]–[6]. The opportunities presented by IoT raise an emerging IoT business model pattern Device-as-a-service (DaaS). We can employ the sensor devices to collect data which would be vended to all interested users or devices. For example, the owner of personal weather station not only uses the IoT devices to control his household heating, but share the data to neighborhood for obtaining financial incentives.

However, most of these IoT devices are easy to be hacked or compromised by various cyber attacks such as a side channel attack. Due to this attack, the secret key in IoT devices might be leaked and then adversaries could successfully forge valid signatures to transfer bitcoins from those devices to other accounts. Actually, ten thefts of over 10 000 BTC each and more than 34 crimes of stealing accidents over 1000 BTC each since 2011 happened [7]. It was reported by Kaspersky labs that about one million infections per month of malware designed to search for secret keys and steal bitcoins were detected [7]. Dubbed Satori IoT Botnet exploits zero-day to zombify Huawei Routers and was found infecting more than 200 000 IP addresses in just 12 h [8]. Cisco's Talos cyber intelligence units discovered an advanced piece of IoT botnet malware, dubbed VPNFilter, that was designed with versatile capabilities to gather intelligence, interfere with Internet communications, as well as conduct destructive cyber attack operations. The malware infected over 500 000 devices in at least 54 countries, most of which are small and home offices routers and Internet-connected storage devices from Linksys, MikroTik, NETGEAR, and TP-Link. Some network-attached storage devices were targeted by the malware as well [9]. To sum up, the security of IoT devices was degraded due to a variety of factors, and the increasing IoT applications based on Bitcoin leads to an urgent need for more secure bitcoin transactions [10].

Blockchain was first introduced by Satoshi Nakamoto in Bitcoin white paper [11]. A blockchain is a hash-based data structure. Each block has a block header, a hash pointer to the previous block, and a Merkle hash tree that digests of the transactions in the block. A blockchain makes use of two well-known cryptographic techniques, namely digital signatures and hash functions. A digital signature is employed to provide the integrity, nonrepudiation, and authentication of bitcoin transactions. A hash function is used to compute a hash value of the previous block and make the blocks as a chain.

Manuscript received June 1, 2018; revised October 8, 2018; accepted October 19, 2018. Date of publication October 29, 2018; date of current version June 19, 2019. This work was supported by the National Key R&D Program of China (2017YFB0802000), National Natural Science Foundation of China (61872229), NSFC Research Fund for International Young Scientists (61750110528), National Cryptography Development Fund during the 13th Five-year Plan Period (MMJJ20170216) and Fundamental Research Funds for the Central Universities (GK201702004, 2018CBLY006). *(Corresponding authors: Yong Yu; Yannan Li.)*

Y. Yu, Y. Ding, Y. Zhao, Y. Li, and Y. Zhao are with the School of Computer Science, Shaanxi Normal University, Xi'an 710062, China (e-mail: yuyong@snnu.edu.cn; liyannan2016@163.com).

X. Du is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA.

M. Guizani is with the Department of Electrical and Computer Engineering, University of Idaho, Moscow, ID 83844 USA.

Digital Object Identifier 10.1109/JIOT.2018.2878406

The decentralization of blockchain benefits IoT in many applications, such as access control to data [12], data trading [13], and key management [14]–[17] in IoT.

A. Related Work

Noyen *et al.* [13] discussed how sensing-as-a-service can benefit from Bitcoin and described the process of exchanging data for cash via Bitcoin. Zhou *et al.* [18] presented distributed data vending on blockchain by combining data embedding and similarity learning. This approach brings the tradeoff between the effectiveness of data retrieval and leakage risk from indexing the data. Leiba *et al.* [19] used blockchain as a decentralized IoT software update delivery network in which participating nodes as distributors are compensated by vendors with digital currency for delivering updates to devices. Delgado-Segura *et al.* [20] introduced a fair protocol for data trading based on Bitcoin script language and double ECDSA. In practical, the script language operator was disabled for Bitcoin transaction. Kopp *et al.* [21] presented KopperCoin, a distributed file storage system with financial incentives. Later, Kopp *et al.* [22] proposed privacy-preserving distributed file storage system with financial incentives, which takes the advantages of ring signatures and one-time addresses to realize a privacy-preserving payment mechanism. However, these solutions provide no confidentiality and reliability of data in the context of side channel attacks.

B. Our Contributions

Blockchain is subject to side-channel attacks due to the openness of its deployment. As a consequence, information leakage especially secret key leakage is possible in various IoT applications. In this paper, we propose a new kind of cryptocurrency, named LRCoin, which is secure even part of the signing key of a user is exposed. LRCoin can be used in the applications where secret key leakage is inevitable, such as the payment of data trading in IoT. We propose a concrete construction of an efficient bilinear pairing-based continual leakage-resilient ECDSA signature algorithm as the building block for signing transactions in LRCoin. The proposed signature algorithm is proven unforgeable against adaptively chosen messages attack in the presence of continual leakage setting. The security proof is conducted in the generic bilinear group model to bypass the impossible results that achieving continual leakage-resilience cryptographic protocols whose secret key is uniquely determined by the corresponding public key. We also implement the proposed signature algorithm on laptops and phones, respectively, which demonstrates that its efficiency is comparable with that of the original ECDSA signature.

II. Data Trading Model in IoT

Data trading [23], [24] is the exchange of bitcoin for data collected by IoT devices between data seller and data buyer. The market is to establish a platform for data seller to use blockchains as infrastructures to sell the data. The data buyer can retrieve data from blockchain and complement the payment. In this section, we introduce the data trading model in IoT and the key components. The participants involved in data trading model in IoT include data sellers, trade market (blockchain network, storage server), and data buyers, as shown in Fig. 1.

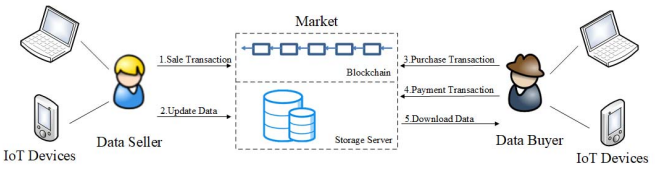


Fig. 1. Data trading model in IoT.

Data Seller: A data seller is a user who owns the data and wants to sell the data.

Trade Market: The trade market is composed of blockchain network and storage server. The blockchain network provides data sellers and data buyers a trading platform. The storage server provides data sellers and data buyers an intermediate facility to upload data and download data, respectively.

Data Buyer: A data buyer submits purchase transactions to the trade market to match the corresponding sale transactions. A data buyer generates a payment transaction for data she wants to buy and downloads the data from the storage sever.

Sale Transaction: A data seller constructs a sale transaction with the topic that he has some data to sell and then broadcasts the transaction to the trade market and uploads the data to the storage server. A sale transaction consists of the topic of the data, the intended price of selling the data, etc.

Purchase Transaction: A data buyer constructs a purchase transaction with the topic of the data that she wants to buy and then broadcasts the transaction to data market. A purchase transaction consists of the topic of the data, the intended price of buying the data, etc.

III. Preliminaries

In this section, we recall some preliminaries used in this paper, including bilinear maps, leakage-resilient models, and the security model for continual leakage-resilient digital signatures.

A. Bilinear Maps

Let $G = <g>, G_T = <g_T>$ be two multiplicative cyclic groups of prime order p with k bits. A map [25] $e : G \times G \rightarrow G_T$ is called a bilinear map if the follow conditions holds.

Bilinearity: For all $u, v \in G$ and $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$.

Nondegeneracy: $e(g, g) \neq 1_{G_T}$, the identity element of G_T.

Efficient Computation: $e(u, v)$ can be computed in polynomial time.

B. Stateful Signatures

In order to achieve continual leakage resilience when a significant bits of secret key are leaked during each round of computations, it is necessary to let the secret key stateful. That is, the secret key must be refreshed after each round of signature computation. Otherwise, the secret key would eventually be completely exposed. Galindo and Vivek [26] suggested to split the secret key into two parts and reserve them on two

distinct parts of a device memory. Specifically, they divide a signature computation into two steps. In each step, the memory in use is divided into two parts called the active part and the passive part. The active part of the memory is the memory being accessed by the computation while other parts of the memory are the passive part. It is assumed that information leakage is only possible in the active part at any specified time.

There are four polynomial-time algorithms, namely KeyGen, Sign_1, Sign_2, and Verify in a stateful signature scheme. Different from the key generation of traditional digital signatures which generates a single secret key, the key generation algorithm in a stateful signature outputs two initial secret states $(S_0, S_0{}')$. The two signing algorithms Sign_1 and Sign_2 are executed sequentially to generate a signature on a message m. A bit more specific, the ith round of the signature computation is performed as follows:

$$\text{Sign}_1(S_{i-1}, m_i, r_i) \rightarrow (S_i, w_i)$$
$$\text{Sign}_2\left(S_{i-1}{}', w_i, r_i{}'\right) \rightarrow \left(S_i{}', \sigma_i\right)$$

where r_i and $r_i{}'$ denote the random values used in Sign_1 and Sign_2, respectively. w_i denotes the state information delivered to Sign_2 by Sign_1. Then the secret state is updated from $(S_{i-1}, S_{i-1}{}')$ to $(S_i, S_i{}')$. The signature of m_i is σ_i.

The formal definition of a stateful signature $\Pi = (\text{KeyGen}, \text{Sign}_1, \text{Sign}_2, \text{Verify})$ is as follows.

1) *KeyGen(k):* On input a security parameter k, it outputs a key pair $(pk, (S_0, S_0{}'))$ where pk is public key and $(S_0, S_0{}')$ are two shares of the secret key.
2) *Sign1(S_{i-1}, m_i):* On input the first part of the $(i-1)$th secret key S_{i-1} and message m_i, it selects a randomness r_i and updates S_{i-1} into S_i and computes the state information w_i, which would be passed onto $\mathbf{Sign}_2$.
3) *Sign2($S_{i-1}{}', w_i$):* On input the second part of the $(i-1)$th secret key $S_{i-1}{}'$ and state information w_i, it chooses a randomness $r_i{}'$ and updates $S_{i-1}{}'$ into $S_i{}'$ and computes the ith signature σ_i.
4) *Verify(pk, σ_i, m_i):* On input the public key pk, signature σ_i, and message m_i, it outputs a bit $b = 1$ meaning valid, or $b = 0$ meaning invalid.

C. Existential Unforgeability With Leakage

A cryptographic primitive is called leakage-resilient if it is secure even the adversary additionally obtains some side-channel information, that is leakage. A variety of leakage models [30]–[32] have been proposed to model side-channel attacks and formalize the security for diverse cryptographic primitives. In 2004, Micali and Reyzin [31] gave a leakage model namely only computation leaks information (OCLI), saying that only the secret memory which is involved in computation at that time leaks information. And they noted that the leakage amount during each computation is bounded. Otherwise, the adversary continually obtains leakage from many computations, and finally the adversary is able to obtain the full knowledge of the secret key. In 2009, Akavia *et al.* [33] introduced a general leakage model called "bounded-memory leakage model." In this model, an adversary is allowed to adaptively choose an efficiently computable leakage function f and send it to a leakage oracle. The adversary obtains $f(sk)$ from the leakage oracle where sk denotes the secret key of the target user. However, the overall output length of all the leakage functions is bounded by a parameter λ, which is smaller than the secret key sk. But this model does not cover the continuous memory leakage, which could been given rise to due to various side channel attacks. In 2010, Brakerski *et al.* [28] and Dodis *et al.* [29] formalized a "continual-memory leakage model." This model is similar to the OCLI model except that in this model the leakage is assumed from the entire secret memory whether or not the memory is involved in computation.

Galindo and Vivek [26] presented an approach to model the leakage in a signature generation by allowing an adversary $\mathcal{A}$ to access to a leakage oracle $\Omega^{\text{leak}}_{\text{secretkey}}(\cdot)$. It not only gives $\mathcal{A}$ signatures of messages choosen by $\mathcal{A}$ but also allows $\mathcal{A}$ to obtain leakage from the current signature computation. More precisely, let λ be the leakage parameter and $\mathcal{A}$ is allowed to adaptively select two efficiently computed leakage functions $f_i() \rightarrow \{0,1\}^{\lambda}$ and $h_i \rightarrow \{0,1\}^{\lambda}$ during every round of signature generation. Specifically, the inputs of leakage functions f_i and h_i are a part of the secret key, respectively, and the outputs of leakage are denoted as $\Lambda_i = f_i(\cdot)$, $\Lambda_i{}' = h_i(\cdot)$. Galindo and Vivek noted that $\mathcal{A}$ can determine h_i after seeing Λ_i. But for simplicity, they only define the leakage model in which f_i and h_i are specified along with the message m_i when they are sent to the leakage oracles.

The property of unforgeability of a stateful signature scheme $\Pi = (\text{KeyGen}, \text{Sign}_1, \text{Sign}_2, \text{Verify})$ with continual leakage is defined by the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1) *Setup:* The challenger $\mathcal{C}$ runs the key generation algorithm $\text{KeyGen}(1^k)$ to obtain a public pair key pk and the initial secret key $(S_0, S_0{}')$. pk is given to $\mathcal{A}$. $\mathcal{C}$ sets a counter $i = 1$ and a set of $\omega = \oslash$ where i denotes the ith round of the signature query and ω is the set of messages which have been signed by querying the Sign-Leak oracle below.
2) *Sign-Leak Queries:* Given with the public parameter pk the adversaries $\mathcal{A}$ can query a Sign-Leak Oracle $A^{\Omega^{\text{leak}}_{S_{i-1}, S_{i-1}{}'}(m_i, f_i, h_i)}(pk)$ at most q numbers of signatures of messages $(m_1, m_2, \ldots, m_i) \in [0,1]^*$ adaptively chosen by $\mathcal{A}$ (i<q). When $\mathcal{A}$ queries the Sign-Leak Oracle, If $|f_i| \neq \lambda$ or $|h_i| \neq \lambda$ the oracle would return $\perp$ and then abort. Otherwise the oracle would response $\mathcal{A}$ with a signature σ_i by computing $\text{Sign}_1(S_{i-1}, m_i) \xrightarrow{r_i} (S_i, w_i)$ and $\text{Sign}_2(S_{i-1}{}', w_i) \xrightarrow{r_i{}'} (S_i{}', \sigma_i)$. During each such signature computation, the adversaries $\mathcal{A}$ could also get some knowledge about the internal secret key from the leakage functions f_i and h_i functioning by $\Lambda_i = f_i(S_{i-1}, r_i)$ and $\Lambda_i{}' = h_i(S_{i-1}{}', r_i{}', w_i)$. After each such query, the counter i is increased to $i+1$ and the messages set is enlarged by $\omega \bigcup m_i$. Eventually the Sign-Leak Oracle returns $(\sigma_i,\ \Lambda_i,\ \Lambda_i{}')$ to the adversary $\mathcal{A}$.
3) *Output:* Finally, $\mathcal{A}$ gives a pair (m, σ). If there exists: a) $\text{Verify}(pk, m, \sigma) = 1$ and b) $m \notin \omega$ then the experiment returns $b = 1$ meaning that $\mathcal{A}$ has won the game.

Otherwise the experiment returns $b = 0$ meaning that $\mathcal{A}$ has failed to forge a signature.

We define $Pr_{\mathcal{A}}^{\text{forge}}$ as the probability of $\mathcal{A}$ wins in the above game. The probability $\Pr_{\mathcal{A}}^{\text{forge}}$ is taken over the coin tosses of $\mathcal{A}$ and KeyGen.

Definition 1: A signature scheme Π is (ϵ, τ, q)-existentially unforgeable under adaptively chosen message attacks with continual leakage if for all (ϵ, τ, q)-adversaries $\mathcal{A}$ where τ is the most running time of $\mathcal{A}$ and $\Pr_{\mathcal{A}}^{\text{forge}}$ is at least ϵ and q is the most number of queries to oracle, the probability $\Pr(b = 1)$ in the experiment Sign-Leak$_{\Pi}(\mathcal{A}, k, \lambda)$ is negligible(as a function of the security parameter k).

D. Generic Bilinear Group Model

The "generic algorithm" was first proposed by Shoup, in which the group elements are encoded as unique binary strings and the special properties of the encodings of the group elements are not exploited. This model was extended to the generic bilinear group [26] where a bilinear map is involved.

The specific details about the generic bilinear group model have been formalized by Galindo and Vivek [26], where representations of bilinear group elements in $\mathbb{G}$ and $\mathbb{G}_T$ are given by random bijective maps $\sigma : \mathbb{Z}_p \to \Xi$ and $\sigma_T : \mathbb{Z}_p \to \Xi_T$. Ξ and Ξ_T are the sets of bit strings, respectively. There are oracles O, O_T, and O_e that can compute the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ and the evaluation of the bilinear map e. These oracles accept the representations in Ξ and Ξ_T as inputs and generate such representations as outputs, which are defined as follows:

$$O(\sigma(a), \sigma(b)) = \sigma(a + b \bmod p)$$
$$O_T(\sigma_T(a), \sigma_T(b)) = \sigma_T(ab \bmod p)$$
$$O_e(\sigma(a), \sigma(b)) = \sigma_T(ab \bmod p).$$

The generator g of the group $\mathbb{G}$ satisfies $g = \sigma(1)$ and the generator g_T of $\mathbb{G}_T$ satisfies $g_T = e(g, g) = \sigma_T(1)$. Since the representation of g is public, thus, users can efficiently generate random elements in both $\mathbb{G}$ and $\mathbb{G}_T$.

We extend the generic bilinear group model [26] slightly to the asymmetric pairing setting denoted as $e : \mathbb{G}_1 \times \mathbb{G}_2 \to G_T$ where $\mathbb{G}_1 \neq \mathbb{G}_2$. Representations of group elements in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are given by random bijective maps $\sigma_1 : \mathbb{Z}_p \to \Xi_1$, $\sigma_2 : \mathbb{Z}_p \to \Xi_2$ and $\sigma_T : \mathbb{Z}_p \to \Xi_T$, respectively. Moreover, the generator P_1 of $\mathbb{G}_1$ satisfies $P_1 = \sigma_1(1)$, and the generator P_2 of $\mathbb{G}_2$ satisfies $P_2 = \sigma_2(1)$, and the generator P_T of $\mathbb{G}_T$ satisfies $P_T = e(P_2, P_1) = \sigma_T(1)$. The adversary $\mathcal{A}$ has accesses to these generic group oracles, namely O_1, O_2, and O_T. Let τ_1, τ_2, and τ_T be the number of queries of $\mathcal{A}$ to group oracles O_1, O_2, and O_T. We define the query form of $\mathcal{A}$ and the response form of generic group oracles as follows. Let the inputs of the ith query of $\mathcal{A}$ to O_T be of the form (X_i, Y_i), where X_i and Y_i are representations in Ξ_T. And let the response of the generic group oracle O_T to the ith query be the representation Z_i, such that $Z_i = X_i \times Y_i$. For convenience, all the above-mentioned representations, including inputs and outputs are denoted as $R_1, R_2, \ldots, R_{3\tau_T}$, meaning that $(X_1, Y_1, Z_1, X_2, Y_2, Z_2 \ldots) = (R_1, R_2, R_3, R_4, R_5, R_6, \ldots)$. The query form of $\mathcal{A}$ to O_1, O_2 and their response form are similar to O_T, except the response of O_1 and O_2 is $Z_i = X_i + Y_i$. Three tables $\mathcal{L}_T$, $\mathcal{L}_1$, and $\mathcal{L}_2$ defined below are maintained to reserve these group element representations obtained from O_1, O_2, and O_T

$$\mathcal{L}_T = \{R_1, R_2, R_3, R_4, R_5, R_6, \ldots, R_{3i} : 1 \leq i \leq \tau_T\}$$
$$\mathcal{L}_1 = \{R_1, R_2, R_3, R_4, R_5, R_6, \ldots, R_{3i} : 1 \leq i \leq \tau_1\}$$
$$\mathcal{L}_2 = \{R_1, R_2, R_3, R_4, R_5, R_6, \ldots, R_{3i} : 1 \leq i \leq \tau_2\}.$$

Since $\mathcal{A}$ can also query group oracles with representations not previously appeared in the above tables, called independent representations, we introduce another three tables in order to maintain the consistences of the representations of the group elements

$$\mathcal{Q}_T = \{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, \ldots, Q_t : 1 \leq t \leq 2\tau_T\}$$
$$\mathcal{Q}_1 = \{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, \ldots, Q_t : 1 \leq t \leq 2\tau_1\}$$
$$\mathcal{Q}_2 = \{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, \ldots, Q_t : 1 \leq t \leq 2\tau_2\}$$

where $Q_1, Q_2, \ldots Q_t$ are received by generic bilinear group oracles in order. It is clear that if the total query number to oracle O_T is τ_T, then $t \leq 2\tau_T$ because the number of independent inputs is at most twice of the number of queries. The same conclusions apply to oracles O_1 and O_2.

Lemma 1: An observer of the interactions between $\mathcal{A}$ and generic bilinear group oracles can determine, for each representations, a sequence of integers, namely $\mathbf{a_i} = (a_{i1}, a_{i2}, \ldots, a_{it})$ with the property that $R_i = \sum_{j=1}^{t} a_{ij} Q_j$ in group $\mathbb{G}_1$ and $\mathbb{G}_2$ or $R_i = \prod_{j=1}^{t} Q_j^{aij}$ in group $\mathbb{G}_T$. We call $\mathbf{a_i}$ the combination of R_i in generic bilinear group $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$.

Brown [27] proved this lemma in a single group $\mathbb{G}$, and it holds naturally in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Thus, Lemma 1 holds.

If there exist two different combinations $\mathbf{a_j} \neq \mathbf{a_k}$ satisfying $R_j = R_k$, we call that a collision appeared in tables $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_T$. When a collision is found in $\mathcal{L}_T$, the observer can conclude

$$\mathbf{u}^{\mathbf{a_j} - \mathbf{a_k}} = \mathbf{1} \bmod \mathbf{p}. \tag{1}$$

Similarly, when a collision appears in $\mathcal{L}_1$ or $\mathcal{L}_2$, the observer has

$$(\mathbf{a_j} - \mathbf{a_k}) \cdot \mathbf{u} = \mathbf{0} \bmod \mathbf{p}. \tag{2}$$

With these equations, the observer is able to infer some knowledge of the set $\mathbf{u}$. Let $r_i \in \mathbb{Z}_p$ be the unique unknown value such that $\sigma_T(r_i) = R_i$, then the observer can obtain some information r_i where $r_i = \mathbf{a_i} \cdot \mathbf{u}$ in $\mathbb{G}_1$ and $\mathbb{G}_2$ or $r_i = \mathbf{u}^{\mathbf{a_i}}$ in $\mathbb{G}_T$.

Lemma 2 [27]: Suppose there are at most m oracle queries, the probability of a collision occurring in a generic bilinear group for Z_p is at most $3\binom{m+1}{2}/p$.

IV. Our Construction

A. Basic Idea

To make the original ECDSA signature scheme continual-leakage-resilient, we apply the techniques due to Galindo and Vivek [26]. That is, instead of managing a single secret key, the secret key is divided into two shares

which are stored in different parts of the memory. The signing algorithm is divided into two steps as well. We deal with the continual leakage of the secret key by refreshing the two shares of the secret key after each signature round regularly to keep the internal secret key stateful.

B. Detailed Construction

The details of the proposed leakage-resilient paring-based ECDSA signature are as follows, in which i denotes the ith round of signing.

Setup(k): On input a security parameter k, this algorithm randomly picks a pairing friendly curve C defined on the finite field $\mathbb{Z}_p$ and outputs the corresponding bilinear pairing generic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 = \mathbb{G}_T$. The operations of $\mathbb{G}_1$ and $\mathbb{G}_T$ are denoted as $+$ and $\times$, respectively. This algorithm chooses a base point P_1 of $\mathbb{G}_1$ and a base point P_2 of $\mathbb{G}_2$, and computes $P_T = e(P_1, P_2)$. $H : \{0,1\}^\star \to \mathbb{Z}_p$ denotes a secure hash function and $f(R) = R \bmod p$ denotes an almost invertible reduction function where $R \in \mathbb{G}_T$ defined in [27]. The system parameters are denoted as para $= (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, P_T, e, p, C, H, f)$.

KeyGen(k, para): On input the security parameter k and the system parameters *para*, this algorithm randomly picks two integers $d, l_0 \in \mathbb{Z}_p$, and computes $S_0 = l_0 P_1$, $S_0{}' = (d - l_0)P_1$ and $Q = P_T^d$. The public key is $pk = (P_1, P_2, P_T, Q)$ while the initial secret key is $sk = (S_0, S_0{}')$.

Sign$_1$(S_{i-1}, m_i): On input the first part of the $(i-1)$th secret key, to sign a message m_i, this algorithm randomly selects integer $l_i, t_i \in [0, p-1]$ and computes $S_i = S_{i-1} + l_i P_1$, $R_s = P_T^{t_i}$, $r_s = f(R_s)$, $h_s = H(m_i)$, and $w_i = t_i P_1 + h_s r_s S_i$.

Sign$_2$(S'_{i-1}, h_s, r_s, w_i): On input the second part of the $(i-1)$th secret key, this algorithm computes $S'_i = S'_{i-1} - l_i P_1$, and $s_i = w_i + h_s r_s S_i{}'$. The signature on the message m_i is $\sigma_i = (r_s, s_i)$.

Verify(pk, m_i, σ_i): On input a message-signature pair (m_i, σ_i) where $\sigma_i = (r_s, s_i)$, this algorithm computes $h_v = H(m_i)$ and $R_v = e(s_i, P_2) \times Q^{-(h_v r_s)}$. Output a bit 1 to indicate the signature is valid if $f(R_v) = r_s$. Otherwise, output 0 meaning the signature is invalid.

V. Security Proof

In this section, we prove the security of our construction against adaptive chosen message attack under continual leakage setting in the generic bilinear group model via a sequence of games. To do this, we first prove the security of our scheme denoted as Π against (Γ-time, 0-query) adversary called a passive adversary under no leakage environment through a game $\mathcal{G}_1$. Then, we prove security against a (Γ-time, q-query) adversary called an active adversary under no leakage setting through a game $\mathcal{G}_2$. It is clear that the attack power of the adversaries in $\mathcal{G}_2$ is stronger than that of in $\mathcal{G}_1$. More precisely, except the generic group oracles O_1, O_2, and O_T, $\mathcal{A}$ can also query a signing oracle to obtain polynomial signatures of messages. Next, we prove security against an active adversary under the continual-leakage setting through a game $\mathcal{G}_3$. In this game, the adversary $\mathcal{A}$ can not only obtain the representations of group elements and signatures of messages but also can get some leaked knowledge of the internal secret key.

A. Proof Under No Leakage Setting

We first provide the unforgeability proof of our scheme against a passive adversary, and then, we give a proof against an active adversary in this part.

Against a Passive Adversary:

Theorem 1: If there exists an $(\epsilon, \tau, 0)$-adversary $\mathcal{A}$ that can forge a valid signature of our scheme Π in the generic bilinear group model, then we can construct an (ϵ', τ')-hash-inverter I_H where

$$\epsilon' \geq \frac{\epsilon - 9\binom{\tau'+1}{2}/p}{\tau'}. \tag{3}$$

Proof: In order to construct a hash inverter I_H, we use the adversary $\mathcal{A}$ as a subroutine. ■

In the generic bilinear group model, the functions σ_T, σ_1, and σ_2 behave randomly. Just like hash functions are controlled by the challenger in the random oracle model, σ_T is controlled by I_H in the generic bilinear group model. With this simulation, we describe the response of I_H to group oracle queries as follows.

Query to O_T: Upon receiving an independent input $R_i = Q_j$ for some i, j from $\mathcal{A}$, the oracle O_T selects a random $u_j \in \mathbb{Z}_p \setminus \{r_1, \ldots, r_{i-1}\}$ and sets $\sigma_T(u_j) = Q_j$. O_T adds Q_j into the table $\mathcal{Q}_T$ and adds u_j to $\mathbf{u}$. If O_T receives a dependent input $R_i = R_k$ for some $k<i$, it sets $r_i = r_k$. Before O_T outputs the response Z_{3i}, it first computes $\mathbf{a_{3i}} = \mathbf{a_i} + \mathbf{a_{i+1}}$ where $\mathbf{a_i}$ and $\mathbf{a_{i+1}}$ can be determined from $\mathcal{A}$'s two inputs. Then it computes $r_{3i} = \prod_{j=1}^{t} u_j^{a_{3i,j}}$. Next, it compares r_{3i} with $\{r_1, \ldots, r_{3i-1}\}$, which are already in the table $\mathcal{L}_T$. If $r_{3i} = r_k$ for some $k<3i$, then O_T responds with R_k. If $r_{3i} \neq r_k$ for all $k<3i$ then O_T selects randomly $R_{3i} \in \Xi_T \setminus \{R_1, \ldots, R_{3i-1}\}$ and appends it into the table $\mathcal{L}_T$.

Query to O_1 and O_2: $\mathcal{A}$'s queries to generic group oracles O_1 and O_2 are similar to queries to O_T. A difference is that O_1 and O_2 computes the preimage as $r_{3i} = \mathbf{a_{3i}} \cdot \mathbf{u} = \sum_{j=1}^{t} a_{3i,j} \cdot Q_j$.

Description of Game $\mathcal{G}_1$: We describe a reduction game from an $(\epsilon, \tau, 0)$-adversary $\mathcal{A}$ to a (ϵ', τ')-hash-inverter I_H of the hash function H. More precisely, if $\mathcal{A}$ can forge a signature in probability ϵ, then I_H can invert H in probability ϵ'.

The input to I_H is a random element $h \in [0, p-1]$ as a challenge, and the goal of I_H is to find M such that $H(M) = h$. To find M, I_H invokes the adversary $\mathcal{A}$ and let it interact with a modified simulation of generic group oracle O_T. When $\mathcal{A}$ is invoked, it makes some queries to generic group oracles. In the end, $\mathcal{A}$ outputs $(M, (r_s, s))$, where M is an arbitrary message and (r_s, s) is a signature for M valid under the public key Q. Initially, we assume without loss generality that the first independent representation in table $\mathcal{Q}_1$ is the base point P_1 in $\mathbb{G}_1$ and $P_1 = Q_1 = \sigma_1(1)$. And the first independent representation in table $\mathcal{Q}_2$ is the base point P_2 in $\mathbb{G}_2$ and $P_2 = Q_1 = \sigma_2(1)$. In the table $\mathcal{Q}_T$, the first independent representation is the signer's public key Q and

$Q = Q_1 = \sigma_T(X)$ where X denotes the discrete logarithm of the signer's secret key.

To make sure the output message M of $\mathcal{A}$ is the answer of the inverter I_H, we introduce the following trick. We conduct a modified simulation of the generic bilinear group oracle O_T. Moreover, we ensure that the modified version of the oracle O_T, from $\mathcal{A}$'s perspective, is indistinguishable from the standard version. Therefore, $\mathcal{A}$ would operate as if it were communicating with a true generic group oracle.

We modify a query from $\mathcal{A}$ to the generic group oracle O_T as follows. Recall that, all queries of $\mathcal{A}$ to O_T have an unique combination $\mathbf{a}$. We consider that the i^{th} query such that $\mathbf{a_{i+1}}$ has the form $(-y, |\mathbf{0})$ where $y \in \mathbb{Z}_p$, and $\mathbf{a_{i+1}} \neq \mathbf{a_l}$ for $l \in [1, i]$ and $\mathbf{a_{3i}} = \mathbf{a_i} + \mathbf{a_{i+1}} \neq \mathbf{a_k}$ for $k \in [1, 3i-1]$. We call this *i special*. For this special i, we modify the way of O_T generating the output R_{3i} as follows. When I_H receives the i^{th} query to O_T, say two inputs (R_i, R_{i+1}), O_T sets

$$R_{3i} = g\left(y \cdot h^{-1} \bmod p\right) \tag{4}$$

where g is the probabilistic inverse of f, which exists since f is almost invertible. Moreover, since h is selected randomly and uniformly from $[0, p-1]$, so is $y \cdot h^{-1} \mod p$. Thus, the distribution of modified simulation R_{3i} is indistinguishable from the uniform distribution over Ξ_T. As a result, $\mathcal{A}$ is able to forge a signature (r_s, s) for message M as it normally would with the true simulation.

Upon receiving the forgery $(M, (r_s, s))$ from $\mathcal{A}$, I_H computes $R_v = e(s, P_2) \times Q^{-(H(M)\cdot r_s)}$ using the method of double-and-multiply among point elements in $\mathbb{G}_T$, with additional n queries to the modified simulation of the generic group oracle O_T. After the last query, I_H gives the response $R_{3(m+n)} = R_v$, which has a unique combination $\mathbf{a_v} = \mathbf{a_i} + \mathbf{a_{i+1}}$.

Note that $R_v = e(s, P_2) \times Q^{-(H(M)\cdot r_s)}$, which is assumed to be the ith query of $\mathcal{A}$ to O_T at this moment. Next we present the response of O_T to this query. Intuitively, there are two possibilities for $\mathbf{a_v}$. If $e(s, P_2)$ is an independent representation then $\mathbf{a}_i = (\mathbf{0}|1)$ and $\mathbf{a_v} = (-H(M) \cdot r_s, 0, 0, \ldots, 1)$. Else if $e(s, P_2) = R_l$ for $l<\tau_T$, then $\mathbf{a_v} = \mathbf{a_l} + (-H(M)\cdot r_s|\mathbf{0}) = (a_{l,1} - H(M)\cdot r_s, a_{l,2}, a_{l,3}, \ldots, a_{l,t})$. If $\mathbf{a_v} \neq \mathbf{a_l}$ for all $l<3(m+n)$ then $R_v = R_{3(m+n)} \neq R_l$ for $l<3(m+n)$. In this case, I_H selects a representation R_v from $\Xi_T \setminus \{R_{T1}, R_{T2}, \ldots R_{3(m+n)-1}\}$. Since A's forgery is valid, so there is $f(R_v) = r_s$. If R_v is selected randomly, the probability of $f(R_v) = r_s$ is at most $3/(p-3(m+n))$. Therefore, we assume $\mathbf{a_{3(m+n)}} = \mathbf{a}_l$ for $l<3(m+n)$. In this case $e(s, P_2)$ is not an independent input. Because if it was independent, the combination $\mathbf{a_v} = (0, H(M)r_s, 0, \ldots, 1)$, which has one more integers than $\mathbf{a}_l$ for $l<3(m+n)$. So $\mathbf{a_i} = \mathbf{a_k}$ for $k<i$ and $\mathbf{a_v} = \mathbf{a_l} = (a_{k,1} - H(M) \cdot r_s, a_{k,2}, a_{k,3}, \ldots, a_{k,t})$. Obviously, $\mathbf{a}_l \neq \mathbf{e_i}$, so R_l is not independent. Therefore, R_l first appeared as an output, say $R_l = R_{3g}$, for $g \in [1, l-2]$. As a result $\mathbf{a_l} = \mathbf{a}_{3g} = \mathbf{a_{3(m+n)}}$. I_H only modifies the response of one of these queries, denoted as j. Moreover this jth query was chosen randomly by I_H before game. So the probability of $j = g$ is at least $1/m$, assuming there has been m queries until now. Because

$$R_v = R_{3j} = g\left(h^{-1} \cdot H(M) \cdot r_s\right)$$

and

$$f(R_v) = h^{-1} \cdot H(M) \cdot r_s = r_s$$

so $H(M) = h \bmod p$ and I_H outputs M as the solution to the given hash inverse problem, such that $H(M) = h$.

The probability ϵ' is bounded as follows. Certainly, we need $\mathcal{A}$ to succeed, which occurs with probability ϵ. Moreover, the above proof assumes no collision occurs, so the probability of collision occurs must be subtracted, which is at most $9\binom{m+n+1}{2}/p$. What is more, the aforementioned proof requires $j = g$, which occurs with probability at least $1/m$. So overall

$$\epsilon' \geq \frac{\epsilon - 9\binom{m+n+1}{2}/p}{m} \geq \frac{\epsilon - 9\binom{\tau}{2}/p}{m} \geq \frac{\epsilon - 9\binom{\tau}{2}/p}{\tau}.$$

Because

$$\frac{\epsilon - 9\binom{\tau}{2}/p}{\tau} \geq \frac{\epsilon - 9\binom{\tau'}{2}/p}{\tau'}$$

we get

$$\epsilon' \geq \frac{\epsilon - 9\binom{\tau'}{2}/p}{\tau'}.$$

Against an Active Adversary:

Theorem 2: If there exists an (ϵ, τ, q)-adversary $\mathcal{A}$ that can forge a valid signature of the proposed scheme Π, then there exists an (ϵ', τ')-collision-finder C_H where

$$\epsilon' \geq \epsilon - 9\binom{\tau'}{2}/p. \tag{5}$$

Proof: The adversary $\mathcal{A}$ can adaptively choose messages to query signatures to C_H. Let ω be the set of messages queried by $\mathcal{A}$. For simplicity, we only describe the differences between the proof of Theorems 1 and 2. ■

Description of Game $\mathcal{G}_2$: In this game, we aim to construct a collision finder C_H using the adversary $\mathcal{A}$. Similarly, $\mathcal{A}$ can interact with a modified simulation of the generic group oracle O_T. The difference is that instead of choosing a random j and modifying the response of the generic group oracle O_T to the jth query, C_H now, for each output Z_{3i} of the generic group oracle O_T, chooses a random message $\tilde{M}_i$, computes $\tilde{h}_i = H(\tilde{M}_i)$ and uses $\tilde{h}_i$ and g to generate a random output in the same way as in the proof of Theorem 1. In this proof, $\mathcal{A}$ can interact with a signing oracle. For each signing query on a message $\hat{M}_i$, C_H selects a random signature $(\hat{r_{si}}, \hat{s_i})$ and $\mathcal{A}$ can query O_T to validate this signature. We stress that O_T must be simulated by C_H in such a way that the queried signatures are valid, using the almost invertibility of f. More precisely, when C_H receives a query message $\hat{M}_i$, it will response $\mathcal{A}$ with a random signature $(\hat{r_{si}}, \hat{s_i})$, and at the same time, C_H computes $e(\hat{s_i}, P_2)$, $Q^{-H(\hat{M}_i)\cdot\hat{r_{si}}}$ and stores them in the table $\mathcal{L}_T$. Furthermore, C_H sets $R_v = e(\hat{s_i}, P_2) \times Q^{-H(\hat{M}_i)\cdot\hat{r_{si}}} = g(\hat{r_{si}})$ and puts it in the table $\mathcal{L}_T$. Then when $\mathcal{A}$ queries O_T with R_v to validate that signature, $\mathcal{A}$ will obtain a value previously computed by C_H. Obviously, this signature is valid with the almost invertibility of function f. On the other hand, when $\mathcal{A}$ queries O_T with special inputs X_i and Y_i, C_H selects a random message $\tilde{M}_i$ and responds A with $g(H(\tilde{M}_i)^{-1} \cdot y)$ where $y \in [1, p-1]$ is the first integer in combination $\mathbf{a}_{Y_i}$. Eventually,

if A outputs a valid forgery (r_s, w) for M, then there exists $f(R_v) = r_s$. We describe the generation of R_v as follows. In the modified simulation, C_H selects a random $\tilde{M}$ and computes $R_v = g(H(\tilde{M})^{-1} \cdot (H(M) \cdot r_s))$. To achieve $f(R_v) = r_s$, there is $H(\tilde{M}) = H(M)$. So overall, under no collision environment, there are two possibilities for $\mathcal{A}$ to forge a valid signature. One is $H(M) = H(\tilde{M})$ and another is $H(M) = H(\hat{M})$. Clearly, by the definition of a forgery, there is $M \neq \hat{M}$. Furthermore, as the distribution of M is uniform and $\tilde{M}$ is selected randomly, so is $M \neq \tilde{M}$. Thus, if $\mathcal{A}$ succeeds, then C_H can find two messages $(M, \tilde{M})$ or $(M, \hat{M})$ with the same hash value. The probability ϵ' is bounded by $\epsilon' \geq \epsilon - 9\binom{\tau'}{2}/p$.

B. Proof Under Leakage Setting

In this proof, we further strengthen the attack power of the adversary $\mathcal{A}$. That is, in addition to group elements, signature queries of adaptively chosen messages, $\mathcal{A}$ can also obtain leakages of secret keys used in computing those signatures.

Theorem 3: Let $\mathcal{A}$ be an (ϵ, τ, q)-adversary that can forge a valid signature of Π, then we can construct an (ϵ', τ')-collision-finder C_H where

$$\epsilon' \geq \epsilon - \frac{\tau'^2}{p} 2^{2\lambda}. \tag{6}$$

Description of Game $\mathcal{G}_3$: In game $\mathcal{G}_2$, the advantage of $\mathcal{A}$ is bounded by its success probability conditioned on the event that no collision has occurred in the lists consisting of elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Note that the proof under nonleakage setting in $\mathcal{G}_2$ and leakage setting in $\mathcal{G}_3$ would be the same conditioned on the fact that a collision has not occurred. The reason is that in the event of no collision, in order to forge a valid signature the adversary has to find a collision of the hash function. Hence, the success probability of $\mathcal{A}$ against scheme Π in and not in leakage setting is same as that in the event of no collision. In the leakage setting, because $\mathcal{A}$ has access to leakage oracles $f_i(sk)$ and $h_i(sk)$ during the ith signature computation, then in adversary's view the secret key is no longer uniformly distributed. Thus, the probability that a collision occurs in the leakage setting is increased by a factor of at most $2^{2\lambda}$. Hence, $\mathcal{A}$ can now cause collisions among representations in tables $\mathcal{L}_T$, $\mathcal{L}_1$, and $\mathcal{L}_2$ with the increased probability. The output of each leakage function is at most λ bits. So overall, there would be at most 2λ bits leaked. So in the view of the adversary, the secret key would only have a min-entropy $\log p - 2\lambda$.

According to Lemma 2, the probability of a collision occurring is increased to

$$\frac{9\binom{m+1}{2}}{p} 2^{2\lambda}. \tag{7}$$

The probability ϵ' of C_H in leakage setting is bounded as

$$\epsilon' \geq \epsilon - \frac{9\binom{m+1}{2}}{p} 2^{2\lambda} \geq \epsilon - \frac{\tau'^2}{p} 2^{2\lambda}. \tag{8}$$

VI. Implementations

In this section, we show the implementations of the proposed leakage-resilient signature algorithm.

TABLE I
Time Cost of Our Signature Scheme

Sub-Algorithm	Setup	KeyGen	Sign	Verification
Laptop	23.268 ms	1.156 ms	11.083 ms	6.083 ms
Phone	40.663 ms	12.380 ms	39.083 ms	14.526 ms

A. Environment

We implemented the proposed scheme on a laptop with 4.00 GB RAM, 64-bit Win 7 operating system and a phone with 4.04 GB RAM, Android 7.1.1 operating system, respectively. The processors of the laptop and phone are Intel Core i5-2450M CPU @ 2.50 GHz and 8 the highest 2.45 GHz, respectively. The implementation was conducted with C++ projects and a powerful Miracl library, which realizes kinds of cryptographic operations such as big integers and elliptic curve group operations. And then compile the C++ projects in Visual Studio 2012.

B. Implementation Results

In the implementation, we choose the security level as AES-80 bit, and employ Cocks-Pinch curve[1] $y^2 = x^3 - 3x + B$, with the embedding degree 2 over a finite field GF(p) where $p = 3 \pmod 4$.

To test the efficiency of our scheme, we executed four subalgorithms Setup, KeyGen, Sign, and Verify, where Sign includes Sign1 and Sign2. Each subalgorithm was run 50 times on the laptop and the phone, respectively. The average running time of the subalgorithms to sign a single message is shown in Table I. The Setup algorithm generates the public parameters of the construction which on average consumes 23.268 ms on the laptop and 40.663 ms on the phone. For KeyGen, it is obvious that the most expensive operations are 2 point multiplications in $\mathbb{G}_1$ and 1 exponentiation in $\mathbb{G}_T$. On average, the KeyGen algorithm costs 1.156 ms on the laptop and 12.380 ms on the phone. In the signature computation, the most expensive operations are 5 point multiplications in $\mathbb{G}_1$, which takes about 11.08 ms on the laptop and 39.09 ms on the phone, respectively, for the Sign1 and Sign2 algorithms together. To validate a signature, the most expensive operations in algorithm *Verify* are 1 bilinear map and 1 exponentiation operation. The *Verify* takes on average 6.083 ms on the laptop and 14.526 ms on the phone.

In the experiment, the implementation results for Setup and KeyGen algorithms are almost constant both on laptop and phone, that is on average 23.268 ms and 1.156 ms on the laptop and 40.663 ms and 12.380 on the phone. This is consistent with the empirical analysis since these two algorithms are independent of the signed messages. We also tested the time cost of the Sign and *Verify* algorithms by increasing the number of signed messages from 1 to 10. The implementation results on the laptop and phone are demonstrated in Figs. 2 and 3. As expected, the time cost of both the Sign algorithm and the *Verify* algorithm increases almost linearly with

[1]Methods for constructing pairing-friendly elliptic curves. [Online]. Available: http://cacr.uwaterloo.ca/conferences/2006/ecc2006/freeman.pdf

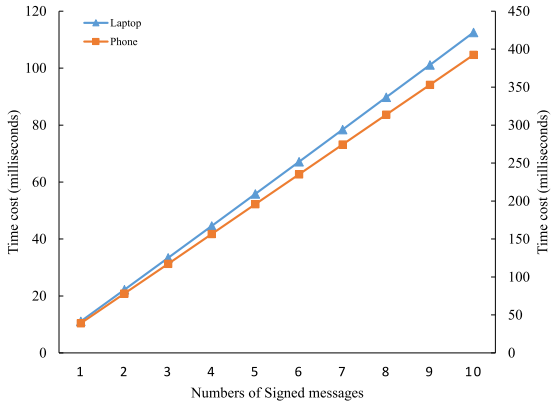


Fig. 2. Time cost of *sign* algorithm.

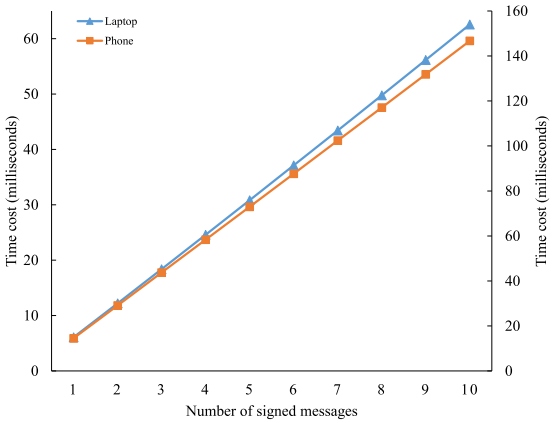


Fig. 3. Time cost of *verify* algorithm.

the increase of the number of signed messages. This is consistent with the theoretical analysis of the proposed scheme too since once a signed message is given, its hash value is determined and all the operations of the Sign and *Verify* algorithms are determined.

VII. Conclusion

Information leakage especially secret key leakage is a serious threat in a number of IoT applications. In this paper, we propose LRCoin, a kind of leakage-resilient cryptocurrency based on Bitcoin, secure even part of the signing key is exposed as a helpful supplement of Bitcoin. LRCoin can be applied to the applications where information leakage is inevitable to make the payment of data trading in IoT more reliable. The core of LRCoin is a leakage-resilient digital signature for signing transactions in the network. We propose a concrete construction of an efficient bilinear-based continual-leakage-resilient ECDSA signature algorithm as the building block of LRCoin. We prove the unforgeability of the proposed signature algorithm in the generic group model. The implementations on the laptop and the phone demonstrate the efficiency and the practicability of our proposal.

References

[1] *More Connected Things is Used*. Accessed: May 1, 2018. [Online]. Available: http://www.gartner.com/newsroom/id/3598917

[2] "Cisco visual networking index: Global mobile data traffic forecast update," San Jose, CA, USA, Cisco Syst., White Paper. Accessed: Apr. 20, 2018. [Online]. Available: http://tinyurl.com/zzo6766

[3] F. Restuccia, S. D'Oro, and T. Melodia, "Securing the Internet of Things: New perspectives and research challenges," *arXiv preprint arXiv:1803.05022*, 2018. [Online]. Available: https://arxiv.org/abs/1803.05022

[4] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *J. Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018. Accessed: Apr. 20, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17315765

[5] Y. Meng *et al.*, "WiVo: Enhancing the security of voice control system via wireless signal in IoT environment," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2018, pp. 81–90.

[6] H. Zhu *et al.*, "You can jam but you cannot hide: Defending against jamming attacks for geo-location database driven spectrum sharing," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 10, pp. 2723–2737, Oct. 2016.

[7] R. Gennaro, S. Goldfeder, and A. Narayanan, "Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security," in *Proc. Int. Conf. Appl. Cryptography Netw. Security*, 2016, pp. 156–174.

[8] *Satori IoT Botnet Zombify Huawei Routers*. Accessed: Jul. 20, 2018. [Online]. Available: https://thehackernews.com/2017/12/satori-mirai-iot-botnet.html

[9] *IoT Botnet Malware Infects Half a Million Routers*. Accessed: Jul. 22, 2018. [Online]. Available: https://thehackernews.com/2018/05/vpnfilter-router-hacking.html

[10] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst.*, Aug. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17318332, doi: 10.1016/j.future.2017.08.020.

[11] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://zoo.cs.yale.edu/classes/cs426/2017/bib/bitcoin.pdf

[12] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," *arXiv preprint arXiv:1506.03471*. [Online]. Available: https://arxiv.org/abs/1506.03471

[13] K. Noyen *et al.*, "When money learns to fly: Towards sensing as a service applications using bitcoin," *arXiv preprint arXiv:1409.5841*, 2014. [Online]. Available: https://arxiv.org/pdf/1409.5841.pdf

[14] X. Du, M. Guizani, Y. Xiao, and H. H. Chen, "A routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 3, pp. 1223–1229, Mar. 2009.

[15] X. Du, Y. Xiao, M. Guizani, and H.-H. Chen, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Netw.*, vol. 5, no. 1, pp. 24–34, Jan. 2007.

[16] Y. Li *et al.*, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Depend. Secure Comput.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/7839231, doi: 10.1109/TDSC.2017.2662216.

[17] Y. Yu *et al.*, "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.

[18] J. Zhou, F. Tang, H. Zhu, N. Nan, and Z. Zhou, "Distributed data vending on blockchain," *arXiv preprint arXiv:1803.05871*, 2018. [Online]. Available: https://arxiv.org/pdf/1803.05871.pdf

[19] O. Leiba *et al.*, "Incentivized delivery network of IoT software updates based on trustless proof-of-distribution," *arXiv preprint arXiv:1805.04282*, 2018. [Online]. Available: https://arxiv.org/pdf/1805.04282.pdf

[20] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomartí, "A fair protocol for data trading based on bitcoin transactions," *J. Future Gener. Comput. Syst.*. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17318344, doi: 10.1016/j.future.2017.08.021.

[21] H. Kopp, C. Bösch, and F. Kargl, "KopperCoin—A distributed file storage with financial incentives," in *Proc. Int. Conf. Inf. Security Pract. Exp.*, 2016, pp. 79–93.

[22] H. Kopp, D. Mödinger, F. Hauck, F. Kargl, and C. Bösch, "Design of a privacy-preserving decentralized file storage with financial incentives," in *Proc. IEEE Eur. Symp. Security Privacy Workshops*, 2017, pp. 14–22.

[23] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 5, pp. 840–852, Sep./Oct. 2018.

[24] F. Liang *et al.*, "A survey on big data market: Pricing, trading and protection," *IEEE Access*, vol. 6, pp. 15132–15154, 2018.

[25] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 213–229.
[26] D. Galindo and S. Vivek, "A leakage-resilient pairing-based variant of the Schnorr signature scheme," in *Proc. IMA Int. Conf. Cryptography Coding*, 2013, pp. 173–192.
[27] D. R. L. Brown, "The exact security of ECDSA," in *Advances in Elliptic Curve Cryptography*, vol. 1363, Rep. CORR 2000-54, pp. 21–40, 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/versions;jsessionid=C373386FDAA1FCF474CC5BA897900E62?doi=10.1.1.32.4312
[28] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan, "Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage," in *Proc. IEEE Found. Comput. Sci.*, 2010, pp. 501–510.
[29] Y. Dodis, K. Haralambiev, A. Lopez-Alt, and D. Wichs, "Cryptography against continuous memory attacks," in *Proc. Found. Comput. Sci. Annu. Symp.*, 2010, pp. 511–520.
[30] T. Malkin, T. Teranishi, Y. Vahlis, and M. Yung, "Signatures resilient to continual leakage on memory and computation," in *Proc. Theory Cryptography Conf.*, 2011, pp. 89–106.
[31] S. Micali and L. Reyzin, "Physically observable cryptography," in *Proc. Theory Cryptography Conf.*, 2004, pp. 278–296.
[32] J. Alwen, Y. Dodis, and D. Wichs, "Leakage-resilient public-key cryptography in the bounded-retrieval model," *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 2009, pp. 36–54.
[33] A. Akavia, S. Goldwasser, and V. Vaikuntanathan, "Simultaneous hardcore bits and cryptography against memory attacks," in *Proc. Theory Cryptography Conf.*, 2009, pp. 474–495.

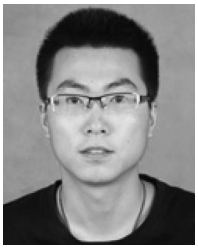
Yong Yu received the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2008.

He is currently a Professor with Shaanxi Normal University, Xi'an. He holds the prestigious One Hundred Talent Professorship of Shaanxi Province. He has authored over 50 referred journal and conference papers. His current research interest includes cryptography and its applications, especially public encryption, digital signature, and secure cloud computing.

Dr. Yu is an Associate Editor of *Soft Computing*.

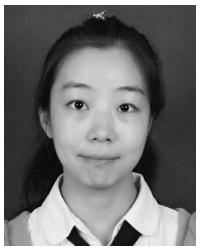
Yujie Ding is currently pursuing the master's degree at the School of Computer Science, Shaanxi Normal University, Xi'an, China.

Her current research interests include digital signatures and blockchain.



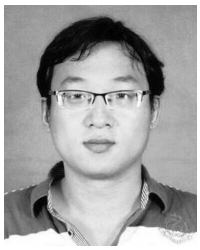
Yanqi Zhao is currently pursuing the Ph.D. degree at the School of Computer Science, Shaanxi Normal University, Xi'an, China.

His current research interests include digital signatures and blockchain.



Yannan Li received the bachelor's and master's degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree at the School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia.

Her current research interests include digital signatures and secure cloud storage.



Yi Zhao is currently pursuing the Ph.D. degree at Shaanxi Normal University, Xi'an, China.

His current research interest includes public key encryption.

Xiaojiang Du (M'04–SM'09) is currently a Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. He has been awarded over $5M in research grants from the U.S. National Science Foundation and Army Research Office. He has authored or co-authored over 200 journals and conference papers. His current research interests include security, systems, wireless networks, and computer networks.

Mohsen Guizani (S'87–M'90–SM'98–F'09) received the B.S. (with distinction) and M.S. degrees in electrical engineering and M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively.

He is currently a Professor and the Electrical and Computer Engineering (ECE) Department Chair with the University of Idaho, Moscow, ID, USA. He has authored 9 books and over 500 publications in refereed journals and conferences. His current research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid.

Dr. Guizani was a recipient of the 2017 IEEE Communications Society Recognition Award for his contribution to outstanding research in wireless communications and three teaching awards and four research awards throughput his career. He is currently the Editor-in-Chief of *IEEE Network Magazine* and serves on the Editorial Boards of several international technical journals and is the founder and the Editor-in-Chief of *Wireless Communications and Mobile Computing* (Wiley). He has guest edited a number of special issues in IEEE publications. He also served as a member, the Chair, and the General Chair of a number of international conferences. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker from 2003 to 2005. He is a Senior Member of the ACM.