Computer Science • 24(1) 2023 https://doi.org/10.7494/csci.2023.24.1.4643

Justice Odoom
Xiaofang Huang
Samuel Danso
Benedicta Nana Esi Nyarko

PRIVACY PRESERVATION FOR TRANSACTION INITIATORS: STRONGER KEY IMAGE RING SIGNATURE AND SMART CONTRACT-BASED FRAMEWORK

Abstract *Recently, blockchain technology has garnered a great deal of support; however, an attenuating factor to its global adoption in certain use cases is privacy-preservation (owing to its inherent transparency). A widely explored cryptographic option to address this challenge has been a ring signature that, aside from its privacy guarantee, must be double-spending resistant. In this paper, we identify and prove a catastrophic flaw for double-spending attacks in a lightweight ring signature scheme and proceed to construct a new fortified commitment scheme that uses a signer's entire private key. Subsequently, we compute a stronger key image to yield a double-spending-resistant signature scheme that is solidly backed by formal proof. Inherent in our solution is a novel, zero-knowledge-based, secure, and cost-effective smart contract for public key aggregation. We test our solution on a private blockchain as well as a Kovan testnet along with a performance analysis that attests to its efficiency and usability – and, we make the code publicly available on GitHub.*

Keywords blockchain, double spending, privacy, ring signature, smart contract, transaction initiator, lightweight

Citation Computer Science 24(1) 2023: 75–96

1. Introduction

Blockchain technology [33, 47] is gradually permeating virtually all spheres of life [1, 2, 41]; this can partly be attributed to the massive ongoing research in diverse areas (as is evident in [22]). It is, however, an indisputable fact that blockchains have inherent challenges such as privacy leaks, scalability, selfish mining, personal identifiable information, and security [17, 52].

Privacy-preservation in blockchain technology is one of the main challenges that has received a considerable level of attention and, consequently, in-depth research by both academia and interested industry experts. This stems from the inherent transparency in blockchains that allows all participants (including adversaries) to access (view) data that is stored on-chain and perform statistical or transaction graph analyses based on which inferences can be established. In Figure 1, we pictorially demonstrate this problem.

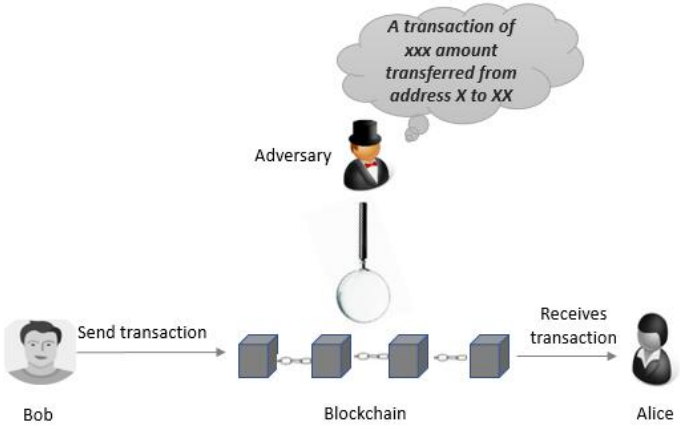

Figure 1. Privacy problem in blockchain

Although blockchain by default provides a level of pseudonymity, it is even possible to pinpoint the real identities of blockchain-based transaction entities by combining on-chain data with off-chain data (as pointed out in numerous works; e.g., [13,32]). In Figure 2, we depict how such deanonymization is possible when a web payment is made via cryptocurrency (as is advanced in [14]). We refer the curious reader to [14] for elucidation on this privacy-breach mechanism, where cookies and trackers play pivotal roles.

It is worth pointing out that privacy-preservation is key to the global adoption of blockchain technology – especially by organizations or businesses, as the disclosure of sensitive information as part of a privacy-breach could have devastating consequences for individuals and corporate organizations (along with possible legal ramifi-

cations). Toward this end, privacy-preserving strategies are actively being researched in academia and industry alike.

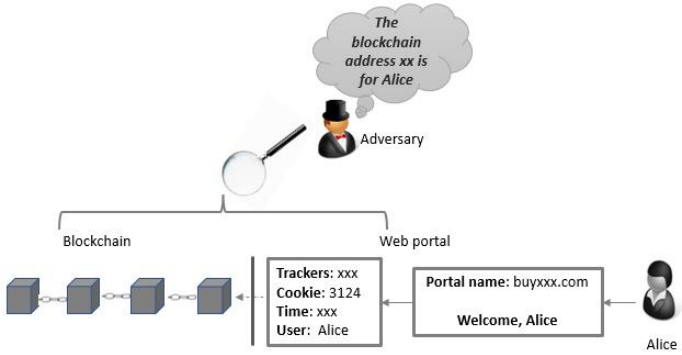

Figure 2. Privacy-breach by combining on-chain and off-chain data

Numerous solutions [8, 18, 21, 30, 46] that encompass such areas as the use of ring signatures, smart contract mixing services, commitments schemes, and zero-knowledge proofs, among others, have been put forward. More recently, differential privacy [19] has been proposed; however, they all have diverse limitations. We refer the reader to a recent survey that elucidates on the aforementioned techniques [42]. It is, however, worth noting that, in practice, only ring signatures have been applied to blockchain technology to preserve privacy [44] by obfuscating the transaction graph. Simply, a ring signature hides the actual signer of a transaction among a group of non-signers, making it is infeasible to infer the actual signer from the output of the signature (thereby guaranteeing signer or transaction-sender anonymity). What is, however, evident is the fact that the construction of a ring signature must prevent the occurrence of double spending.

Double spending is a situation in which a user is able to use the same coin (money) more than once for payments. This is a problem that is quite unique to cryptocurrencies, as digital information is easily reproducible. The catastrophic effects of double spending in the cryptocurrency domain include valueless coins and the overall mistrust or erosion of confidence in the underlying blockchain infrastructure.

2. Existing solutions

In the context of blockchains, privacy entails performing transactions such that no trace or digital footprint can be left that can be tied to an individual's or organization's identity. Diverse mechanisms toward the realization of this requirement have been advanced; we concisely present them in this section.

In [30], the authors described Mobius as a "trustless tumbling for transaction privacy" that is achieved through the use of a linkable ring signature and stealth address. This is described as an Ethereum-based tumbler or what has come to be

known in the blockchain context as a mixing service that often introduces a single point of failure along with other inherent complications [31]. It is, however, refreshing that this laudable idea (embodied in Mobius) is handled by a smart contract; hence, it is autonomous and decentralized. It results in "strong notions of anonymity" (as is described by the authors). Two downsides that are associated with Mobius that are worth pointing out are that, in the first place, the use of Franklin-Zhang's linkable ring signature results in larger signature sizes and retards agile signature verification; hence, it may not be ideal for resource-constrained devices. Second, as admitted by the authors, it is incompatible with the current Ethereum virtual machine (EVM), although a workaround is provided that ultimately results in incentivizing a trustless third party (clearly a disincentive to the transaction recipient by way of giving away a fraction of a transferred ether as an incentive package).

The use of commitment schemes, zero-knowledge proofs, and dual-key stealth addresses to resolve the problem of privacy in terms of creating confidential transactions has also been put forward [45]. This is truly a breakthrough in the context of Ethereum, as it provides confidential transactions hitherto non-existent on the platform (just as Monero [36] and ZCash [5] provide confidential transactions on Bitcoin's platform). However, as admitted by the author, the use of zero-knowledge proofs results in a computational complexity cost-per-transaction (referred to in the Ethereum context as 'gas') of 840,000, which is arguably on the high side. Again, the use of dual-key stealth address protocol limits the feasibility of usage in resource-constrained devices due to the continuous blockchain scanning and the associated computations (which can potentially drain the battery of a device in addition to its time considerations).

Zether [8] also provided a smart contract-based approach along with account balance encryption, allowing users to deposit, transfer, and withdraw funds through zero-knowledge (ZK) proofs with $\sum$-Bullets and ElGamal encryptions, thereby ensuring transactional confidentiality (as in the case of [45]). As laudable as this idea is, it also suffers certain drawbacks. In the first place, Zether is a confidential payment platform; hence, it is limited to Zether- and Ether-based tokens. Again, the size of the ZK-proof for a transfer increases linearly with the size of the anonymity set, resulting in additional costs. Last but not least, Zether is barely feasible (as admitted by the authors), as the cost of a single confidential-only transfer is just below Ethereum's entire block gas limit (the maximum global gas that is consumed per block) of 8 million (attributed to Zether's fairly computation-heavy operations). Even worrying is the fact that, upon completion of the planned modifications that are earmarked in Ethereum's improvement proposals (EIP), the cost of a Zether transfer would be reduced to roughly 1.7 million gas (which is certainly on the high side). Notice that this and the aforementioned techniques (as well as other privacy-preserving techniques) all target cryptocurrencies.

Unlike fiat currencies that are issued by central authorities (banks) that keeps track of the current state of the ownership of money (consequently solving the problem

of double spending), cryptocurrencies counteract the problem without the need for any third party or central authority by using the underlying blockchain's consensus mechanism and key or signature images during the construction of linkable ring signatures [26,29,34,38,39]. However, it is worth pointing out that not all cryptocurrencies are immune to the double-spending problem. In May 2018, Bitcoin Gold (a hard fork of Bitcoin) suffered from the double-spending problem, allowing adversaries to double-spend 388,000 BTG (worth approximately $18 million at that time) [9]. Ethereum classic was also hit in 2019 from which 219,500 ETC (worth approximately $1.1 million) was double spent.

It is therefore evident that flaws in cryptographic protocols and their implementations do occur [20] and are often identified via cryptanalysis. For instance, linkability flaws have been identified in several ring signatures [16, 43]. Au et al. [3] advanced the notion of revocability if linked, consequently deanonymizing a user who generates linked signatures; however, this was later found to be insecure [15], resulting in an improved construction [4]. The case of another ring signature that is based on the discrete logarithm problem also exists [37]; this was later proven to be insecure [51].

2.1. Our contribution

The primary contributions of our paper are summarized as follows:

- We perform cryptanalyis on a lightweight ring signature (LwRS) scheme and show that it is **not** double-spending resistant.
- We construct an improved scheme (dubbed modified lightweight ring signature – *mLwRS*) to guarantee double-spending protection by integrating a stronger key image backed by security proofs.
- We practically demonstrate how to obtain public keys for constructing a ring signature via a smart contract and the proof-of-concept implementation of *mLwRS* together with a performance analysis. The code has been made publicly available on GitHub (along with the full implementation of the smart contract for public key acquisition).

The remainder of this work is organized as follows. In Section 3, we introduce the building blocks that underpin this work, followed by our proposed solution that encapsulates the contributions of this work in Section 4; we then buttress it with a robust security analysis in Section 5. We expatiate on the implementation and testing in Section 6 and provide a thorough evaluation in Section 7. Section 8 concludes the work.

3. Building blocks

This section explains the underlying cryptographic components that were utilized in the proposed solution.

3.1. Key algorithms

Two cryptographic algorithms of interest to us were lightweight ring signature (LwRS) and non-interactive zero-knowledge proofs (NIZKP).

3.1.1. Lightweight Ring Signature (LwRS)

During the course of the ring-signature generation and verification, the use of heavy operations like the pairings and exponentiation that were utilized in [7, 36, 49, 50], among others, resulted in a larger signature size and retarded the signature generation and verification speed.
However, the lightweight ring signature (*LwRS*) [28] that shares the security properties with other ring signatures [12, 23–25, 48] comes bundled with an added merit of suitability for resource-constrained devices in that both the signature generation and verification are very fast and the signature size is drastically reduced. This stems from the fact that computationally expensive operations like pairing is avoided. Instead, multiplication and squaring are utilized.

However, *LwRS* has weak linkability and uniqueness properties owing to a flaw in the construction of the key image, thereby posing a serious flaw in the algorithm. As a result, it is not a counter-measure against the double-spending tendency in the cryptocurrency domain. We prove this and subsequently provide a better construction of the key image in Section 4.2 as an improvement of the *LwRS* scheme. The algorithm for the *LwRS* scheme follows.

Generation of Key Pairs
Given $i = 1, ..., n$ (where n denotes the number of ring members), each i^{th} user selects two safe primes (p_i, q_i) as the private key, where $p_i = 2p'_i + 1$ and $q_i = 2q'_i + 1$.
The public key (N_i) is computed as $N_i = p_i.q_i$. The public key list becomes $\mathcal{L} = (N_1, ..., N_n)$, a defined hash function $\mathcal{H}_i : 0, 1^* \rightarrow Z_{N_i}$ for $i = 1, ..., n$ and a hash function for key images as $\mathcal{H} : 0, 1^* \rightarrow QR(N_i)$, where $QR(N_i) = x \in Z_{N_i}\ s.t.\ x = y^2$ for some $y \in Z_{N_i}$.

Signature generation
Let $\mathcal{L} = (N_1, ..., N_n)$ and (p_g, q_g) denote the public key list for n users and the private key of signer g, respectively. A signature on transaction t by g yields $(\mathcal{L}, t, \sigma)$.

1. The key image I is computed by the g^{th} signer S_g as $\mathcal{H}(p_g||N_g||ID_{event})^{\frac{1}{2}} mod N_g$ using the factorization of N_g (public key) and the Chinese remainder theorem. The ID_{event} (which can be a transaction number or reference/tag) is to allow for multiple transactions that use the same key pair by the signer.
2. S_g randomly selects $r_g \in Z_{N_g}$ and computes $h = \mathcal{H}_1(\mathcal{L}||t||ID_{event})$, and $c_{g+1} = \mathcal{H}_{g+1}(h||r_g)$.
3. Randomly, S_g generates $x_i \in Z_{N_i}$ for all ring members where $i = 1, ..., n$ and $i \neq g$.
4. S_g in turn computes $\forall i : c_{i+1} = \mathcal{H}_{i+1}(h||c_i I + x_i^2)\ mod\ N_i$ sequentially as $g + 1, g + 2, ..., 0, ..., g - 1$.

5. S_g allocates $a_g = r_g - c_g I \, mod \, N_g$ if $r_g - c_g I \, mod \, N_g \in QR(N_g)$ else S_g selects another $x_{g-1} \in Z_{N_{g-1}}$ and then computes a new c_g from the previous step until $r_g - c_g I$ is a quadratic residue.
6. S_g evaluates the $x_g = a_g^{\frac{1}{2}} mod N_g$ courtesy knowledge of the factorization of N_g via the Chinese remainder theorem. Notice that, when using the Tonelli-Shanks algorithm, square roots can equally be solved.
7. S_g produces the signature on transaction t in event ID_{event} as follows:

$$\sigma = (I, c_1, x_1, ..., x_n)$$

Signature verification
The transaction recipient has knowledge of $(\mathcal{L}, \mathcal{H})$; having received signature $\sigma = (I, c_1, x_1, ..., x_n)$ on transaction t during event ID_{event}, he/she proceeds as follows:

1. computes $h = \mathcal{H}(\mathcal{L}||t||ID_{event})$;
2. restores $r_i = c_i I + x_i^2 \, mod \, N_i$ for each $i = 1, ..., n$;
3. calculates $c_{i+1} = \mathcal{H}_{i+1}(h||r_i)$ for each $i = 1, ..., n-1$;
4. σ is accepted ***iff*** output of $c_1 = \mathcal{H}_1(h||r_n)$ is *true* or 1 – thereafter, σ uniqueness is verified by checking that I of signature has not been utilized in past signatures in event ID_{event}; otherwise, it is rejected.

3.1.2. Non-interactive zero-knowledge proofs (NIZKP)

In non-interactive zero-knowledge proofs, Fiat-Shamir heuristics [11] are used to skip back and forth between the interactions of the prover and verifier. Inspired by this and the Schnorr family of signatures [40], we utilize a smart contract that efficiently gets the public key of a user by reducing the heavy computations and eliminating the possible impersonation of the user (since only the user has knowledge of the private key).

In our proposed solution, we implement this via a smart contract (SC) that accepts P, R, e, s as inputs as per the Schnorr signatures from the user and validates $sG \stackrel{?}{=} R + e.P$, where P denotes the public key, R is computed from a random string, e is a challenge that is computed via $\mathcal{H}(R)$, s is the prover's response, and G is the generator point. Note that P, R, e, S are public parameters; hence, they are not secrets. Knowledge of these public parameters (even by adversaries) do not pose any security risk due to the difficulty of the underlying discrete logarithm problem (DLP). This way, the SC acts as a verifier to validate the claim (knowledge of the private key corresponding to P) of the prover before executing the underlying code in a deterministic manner, thereby guaranteeing trusted irrefutable computing that leverages the inherent trait of the transparency of the blockchain technology. The innards of this idea is illustrated in Figure 3.

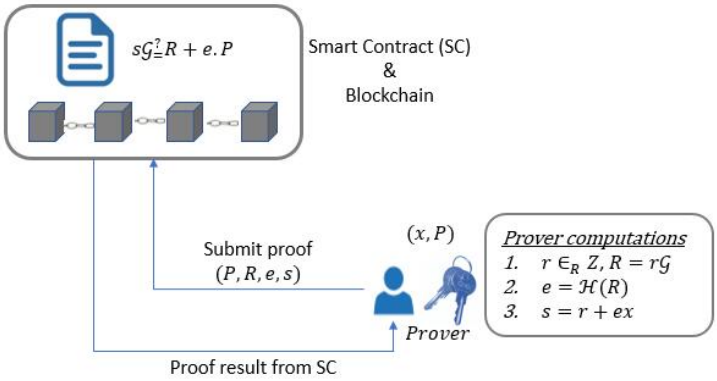

Figure 3. Smart contract-based non-interactive zero-knowledge proof

4. Proposed solution

We propose a modification of the transaction-signing process to implement a double-spending-resistant lightweight ring signature (LwRS). Hereinafter, we refer to our improved (hence, modified) version of *LwRS* as ***mLwRS***.

4.1. Security model

Definition 4.1. A linkable ring signature is a quadruple of the *KeyGen, Sign, Verify,* and *Link* algorithms.

1. **KeyGen**$(1^\lambda) \rightarrow (sk, pk)$: a probabilistic key-generating algorithm in which, upon the input of security parameter λ outputs sk and pk as the user's secret and public keys respectively.
2. **Sign**$(1^\lambda, 1^n, \mathcal{L}, sk, m) \rightarrow \sigma$: a probabilistic signing algorithm in which, upon the input of security parameter λ, the cardinality of the ring n, a list of n public keys $\mathcal{L}$, the signer's secret key sk ***s.t.***, the corresponding public key $pk \in \mathcal{L}$, and the message m to be signed outputs the signature σ.
3. **Verify**$(1^\lambda, 1^n, \mathcal{L}, m, \sigma) \rightarrow 1|0$: a deterministic ring-verification algorithm in which, upon the input of security parameter λ, the cardinality of ring n, a list of n public keys $\mathcal{L}$, and the message signature pair (m, σ) outputs either 1 denoting accept for successful verification or 0 for reject.
4. **Link**$(1^\lambda, 1^n, \mathcal{L}, m_1, m_2, \sigma_1, \sigma_2) \rightarrow (1, 0)$: a Boolean algorithm in which, upon the input of security parameter λ, the cardinality of ring n, list of public keys $\mathcal{L}$, messages m_1, m_2, and signatures σ_1, σ_2 *s.t.* *Verify*$(1^\lambda, 1^n, \mathcal{L}, m_1, \sigma_1) \rightarrow 1$ and *Verify*$(1^\lambda, 1^n, \mathcal{L}, m_2, \sigma_2) \rightarrow 1$, outputs 1 if *linked*; otherwise, 0 if *unlinked*.

Ring signatures must adhere to the security guarantees of signer ambiguity and existential unforgeability (**EUF**).

Definition 4.2 (signer ambiguity). Given $\mathcal{L} = pk_1, ..., pk_n$ along with the corresponding secret keys $\mathcal{L}_{sk} = \{sk_1, ..., sk_n\}$, signer ambiguity holds ***iff*** $\mathcal{L}^*, m^*$ and $\sigma^* \leftarrow \boldsymbol{Sign}(\mathcal{L}^*, m^*, sk_{\varpi})$, where $sk_{\varpi} \in \mathcal{L}_{sk}$, an unbounded adversary hereinafter denoted as $\mathcal{A}$ accepts $\mathcal{L}^*, m^*$, and σ as inputs and outputs ϖ with $Pr[Success] = \frac{1}{n}$. Signer ambiguity therefore postulates the notion of signer anonymity.

Definition 4.3 (existential unforgeability). This security property is modeled as a security game that is played between a challenger (hereinafter $\mathcal{C}$) and $\mathcal{A}$.

1. $\mathcal{C}$ runs ***KeyGen***. Assume $\mathcal{L} = pk_1, ..., pk_n$ with corresponding secret keys $\mathcal{L}_{sk} = \{sk_1, ..., sk_n\}$. $\mathcal{L}$ is made available to $\mathcal{A}$.
2. $\mathcal{A}$ performs adaptive queries on the signing oracle q_s times for valid signatures. Upon the input of any message m and $\mathcal{L}'$, where $\mathcal{L}' \subseteq \mathcal{L}$ with the corresponding secret keys as $\mathcal{L}'_{sk}, \mathcal{C}$ outputs $\sigma \leftarrow \boldsymbol{Sign}(\mathcal{L}', m, sk_{\varpi})$, where $sk_{\varpi} \in \mathcal{L}'_{sk}$ and $1 \leftarrow \mathbf{Verify}(\mathcal{L}', m, \sigma)$.
3. $\mathcal{A}$ guesses a tuple $\mathcal{L}^*, m^*, \sigma^*$ and wins security game ***iff***:
   a) $\mathcal{L}^* \subseteq \mathcal{L}$,
   b) previous queries into the signing oracle were not inclusive of $(\mathcal{L}^*, m^*)$,
   c) $1 \leftarrow \mathbf{Verify}(\mathcal{L}^*, m^*, \sigma^*)$.

Advantage computation: In the above security game, the advantage of $\mathcal{A}$ is computed as

$$Adv(\mathcal{A}) = Pr[\mathcal{A}wins]$$

4.2. Modified LwRS with stronger key image (mLwRS)

The current state of the *LwRS* scheme [28] is not fully immune to the double-spending tendency in the realm of cryptocurrency due to the partial commitment of the private key in the computation of the key image. We show proof of this below:
Given event ID_{eventA}, a user constructs a key image I_A per [28] as

$$I_A = \mathcal{H}(p_g||N_g||ID_{eventA})^{\frac{1}{2}} \; mod \; N_g$$

Merely by swapping p_g (the first part of the signer's private key) in I_A for q_g (the second part of the signer's private key) during the same event ID_{eventA}, a dishonest user can craft another key image I_B as

$$I_B = \mathcal{H}(q_g||N_g||ID_{eventA})^{\frac{1}{2}} \; mod \; N_g$$

Given $I_A \neq I_B$, this facilitates the dishonest user to double-spend (changing the key image, thus allowing the same private key to be used in signing twice on the same event). We demonstrate this in Figure 4, where two different key images can be computed. Note that, owing to the avalanche effect (small change in input results in significant or massive change in digest) that is characteristic of cryptographic hash functions, the change (better still, 'swap') in the private key component as part of the input to the cryptographic hash function outputs a different digest.

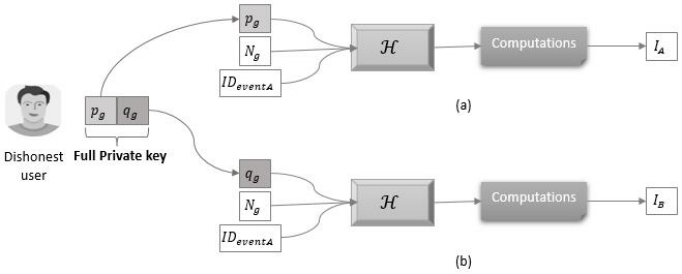

Figure 4. Dishonest user generates two different key images on same event

Consequently, the inherent computations will result in a different key image (I_A in Figure 4a as well as I_B in Figure 4b), thereby arming the dishonest user to double spend. In Figure 5, we clearly demonstrate the possibility of this vulnerability programmatically (the code has been made publicly available – see Section 6) by implementing the original scheme in the event of the key swap by a malicious user as a typical example.

```
sign ==================
found signer j === 0
get h === 7.225895849458249e+76
init random xArr === 4.592528750381499e+153,3.498387388282947e+76,4.442293512117538e+76,6.240679900856024e+75
**********************************
The key image I_A is === 1.58907024941982e+76
**********************************
Simulating malicious user by key swapping
**********************************
The key image I_B is === 3.059636757929099e+76
**********************************
get cArr === 5.902303051787324e+76,3.818716868012821e+75,3.238644809057999e+76,3.186487676206295e+76
get xArr === 6.045338147792888e+76,3.498387388282947e+76,4.442293512117538e+76,6.240679900856024e+75
5.902303051787324e+76
**************************************************************************
Ring Signature generated is:  {
  I: 1.58907024941982e+76,
  c1: 5.902303051787324e+76,
  x: [
    6.045338147792888e+76,
    3.498387388282947e+76,
    4.442293512117538e+76,
    6.240679900856024e+75
  ]
}
**************************************************************************
verify ==================
get h === 7.225895849458249e+76
get xArr === 6.045338147792888e+76,3.498387388282947e+76,4.442293512117538e+76,6.240679900856024e+75
5.902303051787324e+76
get cArr === 5.902303051787324e+76,3.818716868012821e+75,3.238644809057999e+76,3.186487676206295e+76
get rArr === 4.592528750381499e+153,7.728811316089264e+76,4.995516194318694e+76,5.901150327745889e+76
c1 : 5.902303051787324e+76 <===>  c[0]: 5.902303051787324e+76
****************************************
1: Signature verification successful.
****************************************
```

Figure 5. Generation of two different key images on same event by key swapping

The above scenario is possible in that the entire private key pair is not committed during the construction of key image I. We resole this grievous flaw in the original algorithm by proposing ***mLwRS*** with a stronger key image as follows:

1. ***SetUp***: setup two hash functions, $\mathcal{H}$ for key images as well as $\mathcal{H}_1$, where $\mathcal{H}_1 : 0,1^* \rightarrow Z_{N_i}$ for $i = 1, ..., n$, whereas $\mathcal{H} : 0,1^* \rightarrow QR(N_i)$, where $QR(N_i) = x \in Z_{N_i}$ $s.t$ $x = y^2$ for some $y \in Z_{N_i}$.

2. ***KeyGen*** $(\lambda) \rightarrow [(p_i, q_i), N_i]$: for a ring of n members, signer $S_i (1 \leq i \geq n)$ runs ***KeyGen*** to output two safe primes (p_i, q_i) and public key N_i as p_i, q_i. S_i also chooses an ad-hoc ring $\mathcal{L} = (N_1, ..., N_n)$, where $N_i \in \mathcal{L}$.
3. S_i computes a commitment value k followed by a key image I as follows:

$$k = \mathcal{H}_1(p_g + q_g) \; mod \; N_i \tag{1}$$

$$e \leftarrow ID_{event}$$

$$I = \mathcal{H}(k + N_i + e)^{\frac{1}{2}} \; mod \; N_i \tag{2}$$

4. ***Sign(params)***$\rightarrow \sigma$: S_i generates signature σ as $\sigma = (I, c_1, x_1, ..., x_n)$ by following Steps 2 through to 6 of the original signature-generation scheme.
5. Verification of σ by a verifier $\mathcal{V}$ occurs as per the original scheme.

Aside from the safeguard against double spending, this proposed idea clearly protects the privacy of the transaction initiator since the user is embedded within a ring of possible signers. A signature from a ring signature is signer-ambiguous, and the ring members enjoy unconditional anonymity or signer indistinguishability (as reiterated in [7, 25, 30]) and all other ring-signature schemes. To give credence to the fact that our proposed solution thwarts the ability of a malicious user to generate more than one key image via key swapping, we programmatically show the results of our experimentation in Figure 6 (see the link to the GitHub repository in Section 6). We believe this should be sufficient proof of the elimination of the identified vulnerability. Notice that, for the same private key and event tag, the signer can only generate exactly one key image since the same commitment value is computed, thereby forestalling any double-spending tendency via key swapping. We emphasize that our approach adopts a commitment of the entire private key by computing the sum prior to its use (as shown in Equations 1 and 2). This is against the backdrop of the inherent commutativity property of addition.

```
Total user transactions: 2
Signing transaction No.: 1
Using IDevent:  1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The commitment value is:  7.8845239198608 65e+152
********************************
The Stronger key image I is === 4.7778802806223546e+76
Adding Key Image (I) to dataset of key images on the IDevent : 1
Signature verification successful.
********************************
Required number of transactions for mining not met.
Signing transaction No.: 2
Using IDevent:  1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The commitment value is:  7.8845239198608 65e+152
********************************
The Stronger key image I is === 4.7778802806223546e+76
Sorry! Double spending detected. Transaction aborted.
Is the Blockchain valid? true
```

Figure 6. Key-swapping vulnerability solved by key-commitment approach

5. Security and protocol analysis

In what follows, we provide proof that attests to the security of the scheme.

Theorem 5.1 (signature correctness). A valid signature that is constructed by any member from a ring $\mathcal{R}$ yields successful verification, whereas an invalid signature from the same ring fails upon verification.

Proof. Given a verifier that is in possession of the correct public parameters ($\mathcal{L} = (N_1, ..., N_n)$ as a public key list, and cryptographic hash function $\mathcal{H}$), signature verification of the ring signature $\sigma = (I, c_1, x_1, ..., x_n)$ on transaction t proceeds with the restoration of the parameters r_i and c_i for each i from 1 to n and upon verifying that $c_1 = \mathcal{H}(h||r_n)$. Suppose $r_i = c_i I + x_i^2 \, mod \, N_i$ and $c_{g+1} = \mathcal{H}_{g+1}(h||r_g) = \mathcal{H}_{g+1}(h||c_g I + x_g^2) mod N_g$, where $x_g^2 = a_g \, mod \, N_g = r_g - c_g I \, mod \, N_g$ exist within the ring, this yields the following:

$$c_{g+1} = \mathcal{H}_{g+1}(h||c_g I + r_g - c_g I \, mod \, N_g) = \mathcal{H}_{g+1}(h||r_g) \tag{3}$$

The signature correctness satisfies the following condition:

$$\forall(\{p_i, q_i\}, N_i, \mathcal{L}) \leftarrow \textbf{\textit{KeyGen}}, \forall t, \textbf{\textit{Verify}}(\mathcal{L}, t, \sigma) = 1$$

Theorem 5.2 (signature uniqueness). A signer generates only one valid signature on a transaction using the same private public keypair under the same ID_{event}.

Proof. Given a key image I' for the i^{th} user that contains a commitment value k, the key image is computed per the protocol as $I' = \mathcal{H}(k + N_i + e)^{\frac{1}{2}} \, mod \, N_i$.

Following the scheme's algorithm, $\sigma = (I', c_1, x_1, ..., x_n)$. At the verification phase, a randomly selected element $r_i = c_i I' + x_i^2 mod N_i$ is restored for each $i \in [N]$, where n denotes the number of ring members. Any multi-usage of the same σ by $\mathcal{A}$ is detected and discarded if $I_i = I_{i-1}$.

Theorem 5.3 (signature linkability). The scheme is linkable under the same event and is unlinkable otherwise.

Proof. Performing the **Setup** and **KeyGen** algorithms, $\mathcal{C}$ makes $\mathcal{L}, ID_{event}$ and N_i to $\mathcal{A}$ available. Subsequently, $\mathcal{A}$ makes two signature queries to $\mathcal{C}$, and $\mathcal{C}$ outputs two signatures: $(\sigma_1, ID_{event_1}, t_1, \mathcal{L}_1)$, and $(\sigma_2, ID_{event_2}, t_2, \mathcal{L}_2)$ as per the scheme. $\sigma_1 = (I_1, .)$ and $\sigma_2 = (I_2, .)$ are linkable if $I_1 = I_2$, where $I_1 = \mathcal{H}(k_1 + N_1 + ID_{event_1})^{\frac{1}{2}} mod N_1$ and $I_2 = \mathcal{H}(k_2 + N_2 + ID_{event_2})^{\frac{1}{2}} \, mod \, N_2$. As per the scheme, $\mathcal{A}$ must have obtained the K_1 and K_2 values (see Equation 1). This also implies that $ID_{event_1} = ID_{event_2}$.

Theorem 5.4 (signature unforgeability). Based on the factorization problem, it is computationally difficult to generate a valid ring signature without a private key.

Proof. An adversary $\mathcal{A}$ who is not in possession of private key (p_i, q_i) is unable to solve the instance of factorization problem $x_g = a_g^{\frac{1}{2}} \, mod \, N_g$. If forgery success is granted, $\mathcal{A}$ must still compute a valid $c_{i+1} = \mathcal{H}_{i+1}(h||c_i I + x_i^2) \, mod \, N_i$.

Theorem 5.5 (signature ambiguity). Computationally, it is infeasible to deanonymize the signer given a valid signature.

Proof. Given a set of n public keys encapsulated in $\mathcal{L}$, the verifier has a negligible chance of identifying the actual signer. In fact, the probability of a non-ring member identifying the actual signer is computed as

$$Pr[Deanonymization] \leq \frac{1}{n+1}$$

where n denotes the cardinality of the ring.

6. Implementation and testing

The smart contract (refer to Section 3.1.2) for the framework is written in the Solidity language and compiled and tested using Remix IDE – a browser-based tool. It is then deployed on the Kovan testnet. The address of the contract on the Kovan testnet is *0x00ee00443843ff581d081aa181006c27e76cd77b*. Furthermore, we develop a decentralized application (DApp) for the framework, making use of front-end development frameworks and a private blockchain for signature testing. The entire code for this work has been made publicly available via the GitHub link https://github.com/JustNETOrgani/modifiedLwRSonPrvBC.

6.1. Implementation details

As inputs, the prospective user of the framework sends NIZKP public parameters *P, R, e, s* from DApp's front end to the smart contract (SC). A successful proof and transaction execution output a hash of the user's public key $\mathcal{H}(P)$ and three hashed public keys; otherwise, the user is alerted to call *AwaitResponse( )* at another time when the hashed public keys that are stored in the contract satisfy a set threshold (in this case, greater than or equal to 4).

In what follows, we provide two main contract-related methods and omit the others for brevity.

Algorithm 1: AwaitResponse: Get public keys from smart contract for awaiting users

Input: P /*User public key*/
Output: Four hashed public keys
if *user is on awaiting list* **then**
  if $|PubKeyPool| \geq 4$ **then**
    emit four hashed public keys
    change user state to granted public keys
  else
    emit Less public keys available
  end
else
  emit error message
end

Algorithm 2: PartakeInPubKeyPool: Get public keys from smart contract

Input: *P, R, e, s /*Params for NIZK proof.*/*
Output: Four hashed public keys

```
if sG == R + e.P then
    if first time calling SC then
        if |PubKeyPool| ≥ 4 then
            PubKeyPool.push(H(P))
            emit four hashed public keys
            change user state to granted public keys
        else
            Change user state to awaiting
            emit Call AwaitResponse next time
        end
    else
        emit error message
    end
else
    emit failed NIZK proof verification
end
```

Given the fact that the original *LwRS* scheme and the proposed *mLwRS* are not compatible with elliptic curves, the resultant signature cannot be implemented on traditional blockchains like Bitcoin and Ethereum without core protocol modifications. We therefore implement the scheme on a local blockchain as proof-of-concept using JavaScript on a Windows 10 computer with 8GB RAM and a 2.9GHz Intel processor. We provide an overview of our blockchain data structure in Figure 7 that inherits most of the features of classical blockchains but with the signature component being our proposed *mLwRS*.

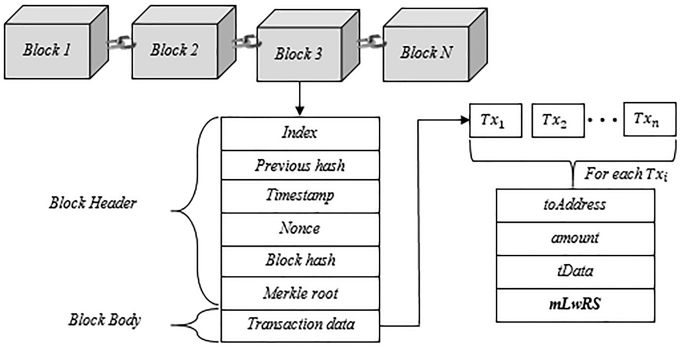

Figure 7. Data structure of our private blockchain

Below, we provide an overview of the algorithms regarding our framework.

1. ***Init(pk)*→ IPFS/IPNS hash**: the framework is initialized by encrypting the framework's public key that uses AES-256 [35] and storing it on the decentralized storage platform interplanetary file system or interplanetary name system (IPFS/IPNS) [6].
2. ***GetUserAccount***$1^{\lambda} \rightarrow (sk, Pk)$: sender acquires private and public key pair on input security parameter.
3. ***PartakeInPubKeyPool***(params)→ $HashedPublicKeys/Await/Reject$: on the aforementioned input NIZKP parameters, the smart contract returns four hashed public keys or places the user on a wait list if the verification of params is successful; otherwise, it returns 'reject.'
4. ***GetPubKeys***(params)→ $PubKeyPool/Reject$: this algorithm is run to decrypt and retrieve public keys that are stored on IPFS/IPNS as part of the framework. It takes the four hashed public keys that were retrieved from the smart contract as params to output an array of raw public keys ($PubKeyPool$) if successful; otherwise, 'reject.'
5. ***Sign***$(\{p_i, q_i\}, PubKeyPool, t) \rightarrow \sigma$: a transaction is signed via $mLwRS$ as shown in the signature-generation process.
6. ***sendTransaction***$(Addr, \sigma, t)$:→ $TnxReceipt/Reject$: the transaction initiator executes this algorithm after signing the transaction; the inputs are destination address ($Addr$), signature σ, and t as the transaction data.

6.2. Testing and validation

We performed the testing by sending a transaction to the $PartakeInPubKeyPool$ contract method, which in turn triggered other contract methods to produce deterministic outcomes. The transaction hash from the Kovan testnet was *0x031fb267f1c3a37a54576e12a97bf5b697eb48ee85a9c715faef0c5a1a8b89a6*, demonstrating the practicability of our smart contract (with Figure 8 showing a sample output from the Kovan testnet).

Figure 8. User gets hashed public keys from contract on Kovan testnet

The four randomly selected hashed public keys that were returned from the contract were what were used by the signer to retrieve the equivalent decrypted raw public keys from IPFS/IPNS (as seen in Figure 9) via **GetPubKeys** to form ad-hoc

ring members during the construction of *mLwRS* to produce a signature/verification output on the local blockchain (as is evident in Figure 10).

Key	Public Key
First	0xe1d7619f12492a3ec630381e41af6dc962fb1b16647859e594a035f67d
Second	0xbf630b2686a85baf74142941e8fc1c4de0ea296f2adbe397259ea73124
Third	0x74eac5db9bac0b4deffbedeb842813b2179893d6985b8ab14efecdc592
Fouth	0x935228b20bdc2349389ade14548ad5542bb16a2df7f3f2e4c293c0319

Figure 9. Decrypted public keys from IPFS/IPNS corresponding to contract's output

```
Total user transactions: 2
Signing transaction No.: 1
Using IDevent:  1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The commitment value is:  7.884523919860865e+152
********************************
The Stronger key image I is === 4.7778802806223546e+76
Adding Key Image (I) to dataset of key images on the IDevent : 1
Signature verification successful.
********************************
Required number of transactions for mining not met.
Signing transaction No.: 2
Using IDevent:  2
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The commitment value is:  7.884523919860865e+152
********************************
The Stronger key image I is === 5.912468165698234e+76
Adding Key Image (I) to dataset of key images on the IDevent : 2
Signature verification successful.
********************************
Pending transactions found. Mining them now...
Hash of block mined: 0000c603f7225d12a8215311969dc6feb6e8b2db1945d06c2945cee811c89b51
Is the Blockchain valid? true
Last Block Info =>: Block index: 2 , Previous hash:  53027800525b032ea618c1db159ef66ea3ad9354dc4225939af3b87d98d94658 ,
Timestamp: 5/10/2021::9:50:57 , Nonce:  363313 , Block hash:  0000c603f7225d12a8215311969dc6feb6e8b2db1945d06c2945cee811c89b51
Entire block: [ { MerkleRoot:
    'a71b26e5e2cc327347279bf60c33b551b4071f04186385317e67f11adb544705',
   BlockBody: { Tx1: [Array], Tx2: [Array] } } ]
Tx1:  [ '{"toAddress":"0x2345a","amount":10,"tData":"LabTest1"}',
  { I: 4.7778802806223546e+76,
    c1: 2.0202877182829137e+76,
    x:
     [ 1.8314019971913142e+76,
       4.4927792843327324e+76,
       3.781600560875915e+75,
       6.363017296028181e+76 ] } ]
```

Figure 10. Transaction signing and verification using mLwRS on local blockchain

7. Evaluation

This section provides a comprehensive evaluation of our proposed scheme.

7.1. Performance evaluation

In Table 1, we measured the performance of our proposed model against the extant schemes. In all cases, the ring size was denoted by N, S denoted the squaring operation, E was the exponentiation, and P was the pairing. Notice from Table 1 that, when compared with the other schemes, our improved scheme had greater efficiency gains at both the signature-generation and verification phases. Even though the signature in [10] was sub-linear, it was without the linkability property that our scheme

enjoyed. Even though our scheme shared same signature-related metrics with [28], note that our scheme was double-spending resistant (hence, ideal) for blockchain and particularly cryptocurrency and e-voting use-cases.

Table 1
Comparative analysis of our framework with our schemes

Evaluation Criteria		Our results	Others	Cite
Signature phase	Sign	$2E + 2M + NS$	$(3 + 4(N-1))E + (1 + 2N)M$	[25]
			$(3 + 2N)E + (2 + 3N)M$	[12]
			$(5 + 6\sqrt{N} + \frac{N+1}{3})E + (6\sqrt{N} + 8)M$	[10]
			$2E + 2M + NS$	[28]
Signature phase	Verify	$M + NS$	$(4N)E + 2NM$	[25]
			$(4N)E + (3N)M$	[12]
			$(6 + 6\sqrt{N})P + (3\sqrt{N} + 1)E + (4\sqrt{N} + 1)M$	[10]
			$M + NS$	[28]
Signature size	Size	$\mathcal{O}(N), N + 2$	$\mathcal{O}(N), N + 2$	[25]
			$\mathcal{O}(N), 2N + 1$	[12]
			$\mathcal{O}(\sqrt{N}), 6\sqrt{N} + 6$	[10]
			$\mathcal{O}(N), N + 2$	[28]
Smart Contract	Public Key storage	32 bytes	64 bytes	[30]
	User traceability	Impossible:Hashed P	Possible:Raw P	[8, 30, 46]
	Prob.(p) of wrong P	$p = 0$ due to NIZKP	$0 < p \leq 1$	[30]
	Gas Cost	214,731	840,000	[30]
			8,000,000	[8]

7.2. Privacy

Our solution provides signer indistinguishability due to the signer anonymity/ambiguity property of *mLwRS*. An honest verifier or even an adversarial blockchain analysis for discovering the actual signer results in a probability of less than $1/n$, where n denotes the total number of ring members. Mindful of the global visibility that characterizes SCs, we store only hashes of the input public keys.

7.3. Security

It is computationally infeasible for an attacker to modify the stored encrypted public keys in that IPFS returns the hash of the content that is stored. Consequently, it guarantees data integrity. As a safeguard measure against possible vulnerabilities in our smart contract, we utilize Oyente [27] – an open-source tool for analyzing smart-contract code against known bugs. Based on Figure 11, our smart contract was free from bugs and, therefore, secure since none of the vulnerabilities that were checked by the auto-auditing tool came out as 'true.'

```
INFO:root:contract testMyScontractFirstTest.sol:NIZkProofs:
INFO:symExec:    ============ Results ===========
INFO:symExec:      EVM Code Coverage:                  23.3%
INFO:symExec:      Integer Underflow:                  False
INFO:symExec:      Integer Overflow:                   False
INFO:symExec:      Parity Multisig Bug 2:              False
INFO:symExec:      Callstack Depth Attack Vulnerability:  False
INFO:symExec:      Transaction-Ordering Dependence (TOD): False
INFO:symExec:      Timestamp Dependency:               False
INFO:symExec:      Re-Entrancy Vulnerability:          False
INFO:symExec:    ====== Analysis Completed ======
INFO:root:contract testMyScontractFirstTest.sol:ZKPComputations:
INFO:symExec:    ============ Results ===========
INFO:symExec:      EVM Code Coverage:                  100.0%
INFO:symExec:      Integer Underflow:                  False
INFO:symExec:      Integer Overflow:                   False
INFO:symExec:      Parity Multisig Bug 2:              False
INFO:symExec:      Callstack Depth Attack Vulnerability:  False
INFO:symExec:      Transaction-Ordering Dependence (TOD): False
INFO:symExec:      Timestamp Dependency:               False
INFO:symExec:      Re-Entrancy Vulnerability:          False
INFO:symExec:    ====== Analysis Completed ======
```

Figure 11. Security report from Oyente

8. Concluding remarks and future works

This work posits a solution that alleviates the privacy concerns of transaction initiators through the use of a privacy-centric zero-knowledge-based smart contract to acquire public keys and subsequently utilize them for generating a modified lightweight ring signature (*mLwRS*) with a stronger key image to withstand a double-spending attack. A smart contract was implemented and deployed on the Kovan testnet as well as a private blockchain in order to test the framework. Aside from the efficiency gains in signature generation and verification, users incur a one-time gas cost of 214,731 for a smart-contract transaction, making it less costly when compared to extant frameworks. Future work would be to make the signature size sub-linear and provide privacy protection for transaction recipients.

Acknowledgements

The research that is presented in this paper was supported by the National Natural Science Foundation of China (Youth), Image Source Forensics Security Research (Grant No. 61702429), and the Sichuan Provincial Department of Science and Technology – a trusted basic service platform for universities based on independent and controllable blockchain technology (Grant No. 2021YFG0311).

References

[1] Abou J.J., Saade R.G.: Blockchain Applications – Usage in Different Domains, *IEEE Access*, vol. 7, pp. 45360–45381, 2019. doi: 10.1109/ACCESS.2019.2902501.

[2] Antonopoulos A.M.: *Mastering Bitcoin: Programming the open blockchain*, O'Reilly Media, Inc., 2017.

[3] Au M.H., Liu J.K., Susilo W., Yuen T.H.: Constant-size ID-based linkable and revocable-iff-linked ring signature. In: *Progress in Cryptology – INDOCRYPT 2006. 7th International Conference on Cryptology in India, Kolkata, India, December 11–13, 2006, Proceedings*, pp. 364–378, Springer, 2006.

[4] Au M.H., Liu J.K., Susilo W., Yuen T.H.: Secure ID-based linkable and revocable-iff-linked ring signature with constant-size construction, *Theoretical Computer Science*, vol. 469, pp. 1–14, 2013.

[5] Ben Sasson E., Chiesa A., Garman C., Green M., Miers I., Tromer E., Virza M.: Zerocash: Decentralized Anonymous Payments from Bitcoin. In: *2014 IEEE Symposium on Security and Privacy*, pp. 459–474, 2014. doi: 10.1109/SP.2014.36.

[6] Benet J.: IPFS-content addressed, versioned, P2P file system, *arXiv preprint arXiv:14073561*, 2014.

[7] Bresson E., Stern J., Szydlo M.: Threshold ring signatures and applications to ad-hoc groups. In: *Advances in Cryptology – CRYPTO 2002, 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002, Proceedings*, pp. 465–480, Springer, 2002.

[8] Bünz B., Agrawal S., Zamani M., Boneh D.: Zether: Towards Privacy in a Smart Contract World. In: J. Bonneau, N. Heninger (eds.), *Financial Cryptography and Data Security. 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers*, pp. 423–443, Springer, Cham, 2020. doi: 10.1007/978-3-030-51280-4_23.

[9] Canellis D.: *Major cryptocurrency exchange delists Bitcoin Gold following $18M hack*, 2018. https://thenextweb.com/hardfork/2018/09/03/bittrex-delists-bitcoin-gold/. [Online; accessed 11 July 2021].

[10] Chandran N., Groth J., Sahai A.: Ring signatures of sub-linear size without random oracles. In: *Automata, Languages and Programming. 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9–13, 2007, Proceedings*, pp. 423–434, Springer, 2007.

[11] Fiat A., Shamir A.: How to prove yourself: Practical solutions to identification and signature problems. In: *Proceedings on Advances in Cryptology – CRYPTO'86*, pp. 186–194, Springer, 1986.

[12] Fujisaki E., Suzuki K.: Traceable ring signature. In: *Public Key Cryptography – PKC 2007. 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16–20, 2007, Proceedings*, pp. 181–200, Springer, 2007.

[13] Gaihre A., Luo Y., Liu H.: Do Bitcoin Users Really Care About Anonymity? An Analysis of the Bitcoin Transaction Graph. In: *2018 IEEE International Conference on Big Data (Big Data)*, pp. 1198–1207, 2018. doi: 10.1109/BigData.2018.8622442.

[14] Goldfeder S., Kalodner H., Reisman D., Narayanan A.: When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies, *Proceedings on Privacy Enhancing Technologies*, vol. 2018, pp. 179–199, 2017. doi: 10.1515/popets-2018-0038.

[15] Jeong I.R., Kwon J.O., Lee D.H.: Analysis of revocable-iff-linked ring signature scheme, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 92(1), pp. 322–325, 2009.

[16] Jia H., Tang C.: Cryptanalysis of a non-interactive deniable ring signature scheme, *International Journal of Information Security*, vol. 20(1), pp. 103–112, 2021.

[17] Joshi A.P., Han M., Wang Y.: A survey on security and privacy issues of blockchain technology, *Mathematical Foundations of Computing*, vol. 1(2), pp. 121–147, 2018. doi: 10.3934/mfc.2018007.

[18] Kosba A., Miller A., Shi E., Wen Z., Papamanthou C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 839–858, IEEE, 2016.

[19] Kus M.C., Levi A.: Investigation and Application of Differential Privacy in Bitcoin, *IEEE Access*, vol. 10, pp. 25534–25554, 2022.

[20] Leung A., Chen L., Mitchell C.: On a Possible Privacy Flaw in Direct Anonymous Attestation (DAA). In: *Trusted Computing – Challenges and Applications. First International Conference on Trusted Computing and Trust in Information Technologies, TRUST 2008 Villach, Austria, March 11–12, 2008 Proceedings*, pp. 179–190, Springer, 2008. doi: 10.1007/978-3-540-68979-9_14.

[21] Li X., Mei Y., Gong J., Xiang F., Sun Z.: A Blockchain Privacy Protection Scheme Based on Ring Signature, *IEEE Access*, vol. 8, pp. 76765–76772, 2020.

[22] Li Y., Marier-Bienvenue T., Perron-Brault A., Wang X., Paré G.: Blockchain technology in business organizations: A scoping review. In: *Proceedings of the 51st Hawaii International Conference on System Sciences*, pp. 4474–4483, 2018.

[23] Liu J.K., Au M.H., Susilo W., Zhou J.: Online/offline ring signature scheme. In: *Information and Communications Security. 11th International Conference, ICICS 2009*, pp. 80–90, Springer, 2009.

[24] Liu J.K., Au M.H., Susilo W., Zhou J.: Linkable ring signature with unconditional anonymity, *IEEE Transactions on Knowledge and Data Engineering*, vol. 26(1), pp. 157–165, 2013.

[25] Liu J.K., Wei V.K., Wong D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: *Information Security and Privacy. 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13–15, 2004, Proceedings*, pp. 325–335, Springer, 2004.

[26] Liu X., Zhang M., Zheng Y., Yang Y.: A linkable Ring Signature Electronic Cash Scheme Based on Blockchain. In: *2020 3rd International Conference on Smart BlockChain (SmartBlock)*, pp. 1–4, 2020. doi: 10.1109/SmartBlock52591.2020.00037.

[27] Luu L., Chu D., Olickel H., Saxena P., Hobor A.: Making smart contracts smarter. In: *CCS'16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 254–269, 2016.

[28] Malina L., Hajny J., Dzurenda P., Ricci S.: Lightweight Ring Signatures for Decentralized Privacy-preserving Transactions. In: *ICETE (2)*, pp. 692–697, 2018.

[29] Mao X., You L., Cao C., Hu G., Hu L.: Linkable Ring Signature Scheme Using Biometric Cryptosystem and NIZK and Its Application, *Security and Communication Networks*, vol. 2021, pp. 1–14, 2021. doi: 10.1155/2021/7266564.

[30] Meiklejohn S., Mercer R.: Möbius: Trustless tumbling for transaction privacy, *Proceedings on Privacy Enhancing Technologies*, vol. 2018(2), pp. 105–121, 2018. doi: 10.1515/popets-2018-0015.

[31] Möser M., Böhme R.: Anonymous Alone? Measuring Bitcoin's Second-Generation Anonymization Techniques. In: *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 32–41, 2017. doi: 10.1109/EuroSPW.2017.48.

[32] Motamed A.P., Bahrak B.: Quantitative analysis of cryptocurrencies transaction graph, *Applied Network Science*, vol. 4, 2019. doi: 10.1007/s41109-019-0249-6.

[33] Nakamoto S.: Bitcoin: A Peer-to-Peer Electronic Cash System. Whitepaper, 2009.

[34] Nassurdine M., Zhang H., Zhang F.: Identity Based Linkable Ring Signature with Logarithmic Size. In: Y. Yu, M. Yung (eds.), *Information Security and Cryptology. 17th International Conference, Inscrypt 2021, Virtual Event, August 12–14, 2021, Revised Selected Papers*, pp. 42–60, Springer, 2021. doi: 10.1007/978-3-030-88323-2_3.

[35] Nechvatal J., Barker E., Bassham L., Burr W., Dworkin M., Foti J., Roback E.: Report on the Development of the Advanced Encryption Standard (AES), *Journal of Research (NIST JRES)*, vol. 3(106), 2001. https://www.nist.gov/publications/report-development-advanced-encryption-standard-aes.

[36] Noether S., Mackenzie A., Lab T.: Ring Confidential Transactions, *Ledger*, vol. 1, pp. 1–18, 2016. doi: 10.5195/LEDGER.2016.34.

[37] Qin M.J., Zhao Y.L., Ma Z.J.: Practical constant-size ring signature, *Journal of Computer Science and Technology*, vol. 33(3), pp. 533–541, 2018.

[38] Ren Y., Guan H., Zhao Q.: An efficient lattice-based linkable ring signature scheme with scalability to multiple layer, *Journal of Ambient Intelligence and Humanized Computing*, pp. 1547–1556, 2022.

[39] Ren Y., Zhao Q., Guan H., Lin Z.: On Design of Single-Layer and Multilayer Code-Based Linkable Ring Signatures, *IEEE Access*, vol. 8, pp. 17854–17862, 2020. doi: 10.1109/ACCESS.2020.2967789.

[40] Schnorr C.P.: Efficient signature generation by smart cards, *Journal of Cryptology*, vol. 4(3), pp. 161–174, 1991.

[41] Sultan K., Ruhi U., Lakhani R.: Conceptualizing blockchains: characteristics & applications, *arXiv preprint arXiv:180603693*, 2018.

[42] Wang D., Zhao J., Wang Y.: A Survey on Privacy Protection of Blockchain: The Technology and Application, *IEEE Access*, vol. 8, pp. 108766–108781, 2020.

[43] Wang H., Zhao S.: Cryptanalysis of Several Linkable Ring Signature Schemes. In: *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 1, pp. 302–305, 2010. doi: 10.1109/NSWCTC.2010.76.

[44] Wang L., Shen X., Li J., Shao J., Yang Y.: Cryptographic primitives in blockchains, *Journal of Network and Computer Applications*, vol. 127, 2018. doi: 10.1016/j.jnca.2018.11.003.

[45] Williamson Z.J.: The Aztec protocol. https://github.com/AztecProtocol/AZTEC/blob/master/AZTEC.pdf. Accessed on 2 July 2021.

[46] Williamson Z.J.: The Aztec protocol, 2018. https://github.com/AztecProtocol/AZTEC.

[47] Wood G.: Ethereum: A secure decentralised generalised transaction ledger, *Ethereum Project Yellow Paper*, vol. 151(2014), pp. 1–32, 2014.

[48] Yang X., Wu W., Liu J.K., Chen X.: Lightweight anonymous authentication for ad hoc group: A ring signature approach. In: *Provable Security. 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24–26, 2015, Proceedings*, pp. 215–226, Springer, 2015.

[49] Yuen T.H., Liu J.K., Au M.H., Susilo W., Zhou J.: Efficient linkable and/or threshold ring signature without random oracles, *The Computer Journal*, vol. 56(4), pp. 407–421, 2013.

[50] Zhang F., Kim K.: ID-based blind signature and ring signature from pairings. In: *Advances in Cryptology – ASIACRYPT 2002. 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1–5, 2002, Proceedings*, pp. 533–547, Springer, 2002.

[51] Zhang J., Bai W., Jiang Z.: On the Security of a Practical Constant-Size Ring Signature Scheme, *International Journal of Network Security*, vol. 22(3), pp. 394–398, 2020.

[52] Zheng Z., Xie S., Dai H., Chen X., Wang H.: An overview of blockchain technology: Architecture, consensus, and future trends. In: *2017 IEEE International Congress on Big Data (BigData congress)*, pp. 557–564, 2017.

Affiliations

Justice Odoom
Southwest University of Science and Technology, Department of Computer Science and Technology, Mianyang 621010 Sichuan China, odoom.justice@ieee.org

Xiaofang Huang
Southwest University of Science and Technology, Department of Computer Science and Technology, Mianyang 621010 Sichuan China, huangxiaofang@swust.edu.cn

Samuel Danso
Ghana Communication Technology University, Faculty of Engineering, Accra, PMB 100, Ghana, sdanso@gctu.edu.gh

Benedicta Nana Esi Nyarko
Southwest University of Science and Technology, Department of Information Engineering, Mianyang 621010 Sichuan China, benedictanyarko41@gmail.com

Received: 16.01.2022
Revised: 13.12.2022
Accepted: 16.12.2022