Qin MJ, Zhao YL, Ma ZJ. Practical constant-size ring signature. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(3): 533–541 May 2018. DOI 10.1007/s11390-018-1838-z

Practical Constant-Size Ring Signature

Meng-Jun Qin, Yun-Lei Zhao*, *Member*, *CCF*, and Zhou-Jun Ma

Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai 201203, China
State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710126, China
State Key Laboratory of Cryptology, State Cryptology Administration, Beijing 100878, China

E-mail: {mjqin16, ylzhao, zjma14}@fudan.edu.cn

Received October 31, 2017; revised March 28, 2018.

Abstract Bitcoin has gained its popularity for almost 10 years as a "secure and anonymous digital currency". However, according to several recent researches, we know that it can only provide pseudonymity rather than real anonymity, and privacy has been one of the main concerns in the system similar to Bitcoin. Ring signature is a good method for those users who need better anonymity in cryptocurrency. It was first proposed by Rivest *et al.* based upon the discrete logarithm problem (DLP) assumption in 2006, which allows a user to sign a message anonymously on behalf of a group of users even without their coordination. The size of ring signature is one of the dominating parameters, and constant-size ring signature (where signature size is independent of the ring size) is much desirable. Otherwise, when the ring size is large, the resultant ring signature becomes unbearable for power limited devices or leads to heavy burden over the communication network. Though being extensively studied, currently there are only two approaches for constant-size ring signature. Achieving practical constant-size ring signature is a long-standing open problem since its introduction. In this work, we solve this open question. We present a new constant-size ring signature scheme based on bilinear pairing and accumulator, which is provably secure under the random oracle (RO) model. To the best of our knowledge, it stands for the most practical ring signature up to now.

Keywords ring signature, constant size, bilinear pairing, accumulator

1 Introduction

Since Bitcoin was first proposed by Satoshi in 2008[1], as the basic technology of Bitcoin, blockchain has gained a lot of popularity. However, in most blockchain applications, anonymity could not be guaranteed, and applications like Bitcoin can only provide pseudonymity[2] rather than real anonymity. There are lots of researchers that study the bitcoin anonymity[2-4] and try to track the person in the real world who starts the transaction; thus how to keep anonymous in a transaction has been one of the main concerns in the system like Bitcoin. Fortunately, ring signature can make it possible.

Ring signature scheme was first proposed by Rivest *et al.*[5] It allows a user to sign a message on behalf of a group of users even without their coordination. Different from group signature[6], ring signature scheme has no reliable third party, which is called the group manager who controls all the privacy information about the real signer in the group. With ring signature, when a user (e.g., Bob) wants to sign a message without revealing his identity, he can choose a set of potential signers including himself to protect his own identity. These potential signers are chosen arbitrarily, and may even not know that they are included in some ring. However, from the signature, everyone appears to be the real signer with the same probability, which ensures the anonymity of ring signature.

Regular Paper

Special Section on Blockchain and Cryptocurrency Systems

This work is supported in part by the National Key Research and Development Program of China under Grant No. 2017YFB0802000, the National Natural Science Foundation of China under Grant Nos. 61472084 and U1536205, the Shanghai Innovation Action Project under Grant No. 16DZ1100200, the Shanghai Science and Technology Development Funds under Grant No. 6JC1400801, and the Shandong Provincial Key Research and Development Program of China under Grant No. 2017CXG0701.

*Corresponding Author

With the development of information technology, privacy has been one of the most important concerns in people's daily life. Since ring signature can protect the signer's privacy, it is used in many interesting applications. The first application is providing a way of leaking a message secretly, as described in [5], so that the receiver can make sure that the message is actually generated by some official source, and at the same time the person who leaks the message cannot be distinguished from all possible persons who appear in the ring signature. Another popular application is providing untrackability[7] in a digital cryptocurrency, named Monero[8]. Here, roughly speaking, untrackability means that for each incoming transaction, all possible signers have the same probability of issuing this transaction. In this way, the real sender of the transaction is protected from being tracked. In addition, ring signature also has applications in designated verifier signatures[9], e-voting[10], and more[11-12].

Readers who want to know the details of how ring signature works in cryptocurrency may refer to [8].

1.1 Related Work

For ease of reference, the existing ring signature schemes are briefly summarized in Table 1.

Rivest *et al.*[5] proposed the first ring signature scheme, based upon the discrete logarithm problem (DLP) assumption, by constructing a circle equation using trapdoor permutation. Many of the following schemes[13-14] followed this idea but with different techniques to form the circle equation.

In [15], Boneh *et al.* first used bilinear pairings in constructing ring signature schemes. The construction proposed in [16] uses ZAP as the building tool, which is a 2-round, public coin, witness-indistinguishable proof system for the NP (non-deterministic polynomial) problem. For most of the existing ring signature schemes[5,13-16], the signature size grows linearly with the ring size. Therefore, when the ring size is large, the resultant ring signature becomes unbearable for some power limited devices or leads to heavy burden over the communication network. The first sub-linear size ring signature was introduced by Chandran *et al.*[17] under the assumptions of strong-Diffie-Hellman (SDH) and subgroup decision problem (SGD), and then followed by Yuen *et al.*[18] and Ghadafi[19]. The scheme proposed in [19] is based on a relatively non-standard assumption called q asymmetric double hidden strong Diffie-Hellman (q-ADH-SDH). In these schemes, the public keys in the ring are represented by a matrix, and then non-interactive witness indistinguishable proofs are used to prove the set memberships.

To the best of our knowledge, there are only two ring signature schemes whose signature size is independent of the size of the ring. The first one was proposed by Dodis *et al.*[11] based on anonymous identification in ad hoc groups, with efficient implementations based upon the strong RSA assumption in the random oracle model via Fiat-Shamir transformation. But the strong RSA based instantiation in [11] uses quite complex Σ-protocols, and may not be practical enough.

The latest approach was recently proposed in [21] by using a generic constant-size set member proof. But their instantiation is based on the q-strong Diffie-Hellman (q-SDH) assumption and the symmetric external Diffie Hellman (SXDH) assumption via the Boneh-Boyen signature scheme, which makes it less efficient in practice. To the best of our knowledge, implementing a practical constant-size ring signature based on the DLP assumption remains open.

Table 1. Overview of Some Existing Ring Signature Schemes

Reference	Size	Assumption	Model	Note
Rivest *et al.*[5]	$O(n)$	DLP	RO	Circle equation using trapdoor permutation
Abe *et al.*[13]	$O(n)$	DLP	RO	Hash and trapdoor one-way function
Liu *et al.*[14]	$O(n)$	DDH	RO	Circle equation using hash
Boneh *et al.*[15]	$O(n)$	Co-CDH	RO	Bilinear pairings
Bender *et al.*[16]	$O(n)$	PKE, ZAP	Standard	Generic construction
Chandran *et al.*[17]	$O(\sqrt{n})$	SDH & SGD	Standard	Proof for set membership commitment
Yuen *et al.*[18]	$O(\sqrt{n})$	SDH & SGD	Standard	Linkable and/or threshold
Ghadafi *et al.*[19]	$O(\sqrt{n})$	q-ADH-SDH[20]	Standard	Sub-linear size set membership proof
Dodis *et al.*[11]	$O(1)$	Strong-RSA	RO	Ad-hoc anonymous identification, relatively complex Σ-protocols
Bose *et al.*[21]	$O(1)$	q-SDH & SXDH	Standard	Generic constant-size set membership proof
Ours	$O(1)$	DLP	RO	Accumulator, and bilinear pairings

Note: RO: Random Oracle. DDH: Decisional Diffie-Hellman. Co-CDH: Co-Computational Diffie-Hellman.

1.2 Our Contribution

In this work, we present a new constant-size ring signature scheme based on bilinear pairing and accumulator, which is provably secure under the DLP assumption in the random oracle model. To the best of our knowledge, this is the first constant-size ring signature scheme based on the DLP assumption, and stands for the most practical ring signature up to now. In particular, compared with the existing constant-size ring signature schemes, our scheme is simpler and more efficient, making it more suitable for large size ring and easy for implementation in practice. Along the way, we make a detailed comparative study on some of the existing ring signature schemes in the literature.

2 Preliminaries

Notations. Let N be the set of positive integers. We denote by Z_q an additive group of integers of prime order q. A string means a binary one which is $\{0,1\}^*$. The empty string is denoted as ϵ. If $x_1, x_2, ..., x_n$ are objects, then $x1||x2||...||x_n$ denotes an encoding of them as strings (e.g., concatenation) from which the constituent objects are easily recoverable. If S is a set, $|S|$ means the number of elements in the set, and $s \leftarrow_R S$ denotes the operation of choosing an element s from S uniformly at random. If b is an integer, then $a \leftarrow b$ means $a = b$. If A is a randomized algorithm, then $A(x_1, ..., x_n; \rho)$ denotes its output on inputs $x_1, ..., x_n$ and random coins ρ, while $y \leftarrow_R A(x_1, ..., x_n)$ means that we choose ρ at random and let $y = A(x_1, ..., x_n; \rho)$. Assume S, C are two sets, $S \setminus C$ denotes removing all the elements that appear in C from S. A PT algorithm means the algorithm can be executed in polynomial time, and PPT denotes probabilistic polynomial time. $Pr[E : R_1, ..., R_n]$ denotes the probability of event E after the sequential execution of random processes $R_1, ..., R_n$.

2.1 Bilinear Pairing

A bilinear pairing is defined to be $\mathcal{G} = (p, q, G_1, G_2, G_T, e, g_1, g_2)$, where $G_1 = g_1$, $G_2 = g_2$ and G_T are multiplicative groups of prime order p. Let $e : G_1 \times G_2 \to G_T$ be a map with the following properties.

• *Bilinear.* $\forall g_1 \in G_1, g_2 \in G_2$ and $a, b \in Z_q$: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.

• *Non-Degenerate.* There exists $u \in G_1, v \in G_2$ such that $e(u, v) \neq \mathcal{O}$, where $\mathcal{O}$ stands for the identity element of the group.

• *Computability.* There is an efficient algorithm to compute $e(u, v)$ for all $u \in G_1, v \in G_2$.

There are three main types of pairings according to the literature[9,22].

• *Type*-1. The groups G_1, G_2 are the same.

• *Type*-2. $G_1 \neq G_2$, but there exists an isomorphic $\zeta : G_2 \to G_1$ that can compute G_1 from G_2 efficiently.

• *Type*-3. $G_1 \neq G_2$, and there are no efficiently computable homomorphisms between G_1 and G_2.

For simplicity, we will use type-1 bilinear pairings in our construction, but it should be noticed that type-2 or type-3 also works for our construction. Because no matter whether the exponent is in the position of g_1 or g_2, the result keeps the same.

2.2 Accumulator

Definition 1. *An accumulator is a tuple $(\{X_\lambda, F_\lambda\})$, where $\lambda \in N$, $\{X_\lambda\}$ is called the value domain of the accumulator, and $\{F_\lambda\}$ is a collection of pairs of functions such that each $(f, y) \in F_\lambda$ is defined as $f : U_f \times X_f \to U_f$ for some $X_f^{ext} \supseteq X_\lambda$, and $y : U_f \to U_y$ is a bijective function where U_f and U_y denote the value domain of functions f and y respectively. In addition, the following properties are satisfied.*

• *Efficient Generation. There exists an efficient algorithm that takes as input a security parameter 1^λ and outputs a random element $(f, y) \in_R F_\lambda$, possibly together with some auxiliary information. And in the following sections, we denote the algorithm by ACC.Gen.*

• *Quasi-Commutativity. For every $\lambda \in N, (f, y) \in F_\lambda, u \in U_f, x_1, x_2 \in X_\lambda : f(f(u, x_1), x_2) = f(f(u, x_2), x_1)$. For any $\lambda \in N, (f, y) \in F_\lambda$ and $X = \{x_1, x_2, ..., x_n\} \subset X_\lambda$, we call $y(f(f(u, x_1), ..., x_n))$ the accumulated value of the set X over u. Due to quasi-commutativity, the value $f(f(u, x_1), ..., x_n)$ is independent of the order of x_i and is denoted by $f(u, X)$.*

• *Efficient Evaluation. For every $(f, y) \in F_\lambda, u \in U_f$ and $X \subset X_\lambda$ with polynomially-bound size: $y(f(u, X))$ is computable in time polynomial in λ, and we use ACC.Eval to represent the process of computing the accumulated value. Also, there is a witness w meaning that some variable x has been accumulated within $v = f(u, X)$ iff $f(w, x) = v$, and we use ACC.Wit to denote computing the witness w.*

In this paper, we use the same accumulator as in [23], where the function pair (f, y) is defined as $f : Z_q \times Z_q \to Z_q, y : Z_q \to Z_q$, namely, $f : (\beta, x) \to \beta(x + d)$, where $d \leftarrow_R Z_q$ and $\beta \in_R Z_q$, and y is the identity function $y(x) = x$. By this accumulator, for any $X = \{x_1, ..., x_n\} \subset Z_q$, a typical form is that $f(\beta, X) = \beta(x_1 + d)...(x_n + d)$, and it is obvious that all the above three properties in Definition 1 are satisfied (the reader is referred to [23] for proof details). It is easy to see that the main drawback of our accumulator is that it needs a trusted third party to compute both accumulated value and witness, but as described in [24], we can also use multiparty computation technique to solve this problem.

2.3 General Forking Lemma

In order to prove that our constant-size ring signature scheme satisfies unforgeability under the insider corruption, we use a generalization of the Forking Lemma[25] proposed by Bellare and Neven.

Lemma 1 (General Forking Lemma[25]). *Fix an integer $\gamma \geqslant 1$ and a set H of size $|H| \geqslant 2$. Let A be a randomized algorithm that takes $x, h_1, h_2, ..., h_\gamma$ as input and returns a pair (J, σ) where $J \in \{0, ..., \gamma\}$ and σ is referred to as a side output. Let IG be a randomized algorithm that we call the input generator. The accepting probability of A is defined as $acc = Pr[J \geqslant 1 : x \leftarrow_R IG, h_1, ..., h_\gamma \leftarrow_R H; (J, \sigma) \leftarrow_R A(x, h_1, ..., h_\gamma)]$. The forking algorithm F_A (Algorithm 1) associated to A is the randomized algorithm that takes x as the input and proceeds as follows.*

Algorithm 1. $F_A(x)$

```
1:  Pick coins $\rho$ for $A$ at random
2:  $h_1, ..., h_\gamma \leftarrow_R H$
3:  $(J, \sigma) \leftarrow A(x, h_1, ..., h_\gamma; \rho)$
4:  if $J = 0$ then
5:      return $(0, \epsilon, \epsilon)$
6:  endif
7:  $h'_J, ..., h'_\gamma \leftarrow_R H$
8:  $(J', \sigma') \leftarrow A(x, h_1, ..., h_{J-1}, h'_J, ..., h'_\gamma; \rho)$
9:  if ($J = J'$ and $h_J \neq h'_J$) then
10:     return $(1, \sigma, \sigma')$
11: else
12:     return $(0, \epsilon, \epsilon)$
13: endif
```

Let $frk = Pr[b = 1 : x \leftarrow_R IG; (b, \sigma, \sigma') \leftarrow_R F_A(x)]$, then

$$frk \geqslant acc(\frac{acc}{\gamma} - \frac{1}{|H|}). \quad (1)$$

The complete proof of this lemma can be found in [25]. The general idea is that if randomized algorithm A accepts with probability α, then a rewind of A accepts with the probability roughly α^2. Besides, we should notice that the same random coins are used in both the first and the execution of A.

3 Ring Signature Definitions

Definition 2. *A ring signature scheme is a tuple (Gen, Sign, Verify) of PT algorithms, where each of them means generating a key pair, signing a message, and verifying the signature for the message using the corresponding public keys, respectively. Formally they are described as follows.*

• *$Gen(1^l)$. It takes as input a security parameter 1^l, and returns a public key PK also called the verification key and a private key SK also known as the signing key.*

• *$Sign(SK, m, R)$. The signer outputs a signature σ on a message m with respect to a ring R using the signing key SK. We assume that the number of public keys in the ring $|R| \geqslant 2$, and there is exactly one public key in R corresponding to the signing key SK, and all the keys in R are generated by Gen.*

• *$Verify(\sigma, m, R)$. The verifier can be anyone, including the adversary, who verifies the signature σ on a message m with respect to the ring R. If the verifier accepts the signature, then the algorithm returns 1; otherwise, returns 0.*

With the above definition, we can now describe how a ring signature scheme works: at the beginning, each of a collection of users runs $Gen(1^l)$ to generate a pair of keys (PK, SK) under the same security parameter l. When a user (e.g., Bob) wants to sign a message m anonymously, he chooses a group of public keys R, which can be done even without the coordination of the owners of the public keys in R. Then Bob puts his public key into R randomly, runs $Sign(SK, m, R)$, and outputs a signature (σ, R). At this time, from the view of other users, each public key in R has essentially the same possibility of generating the signature, and anyone can verify the signature by running $Verify(\sigma, m, R)$.

Definition 3. *A ring signature scheme satisfies completeness, if for every security parameter l, every (PK, SK) outputted by $Gen(1^l)$, and every message $m \in \{0,1\}^*$, it holds that*

$$Verify(Sign(SK, m, R), m, R) = 1,$$

where $PK \in R$.

Anonymity is a special property that ring signature enjoys, and it is this property that makes ring signature much desirable and widely-used in privacy-preserving applications in practice. Here, we use a similar anonymity definition from [17].

Definition 4. *A ring signature scheme $(Gen, Sign, Verify)$ has perfect anonymity, if a signature on message m under public key $PK_{i_0} \in R$ and* *a signature on message m under public key $PK_{i_1} \in R$ have the same distribution, where $i_0 \neq i_1$, R is the ring and the same for both signatures. This means that the signer's key is hidden among all the honestly generated keys in the ring. Formally, we require that, for any PPT adversary $\mathcal{A}$ and security parameter l, it holds*:

$$Pr[\mathcal{A}(\sigma) = 1 : (m, i_0, i_1, R) \leftarrow \mathcal{A}^{Gen(1^l)};$$
$$\sigma \leftarrow Sign(SK_{i_0}, m, R)] = Pr[\mathcal{A}(\sigma) = 1 :$$
$$(m, i_0, i_1, R) \leftarrow \mathcal{A}^{Gen(1^l)}; \sigma \leftarrow Sign(SK_{i_1}, m, R)],$$

where $\mathcal{A}$ selects i_0, i_1 such that (PK_{i_0}, SK_{i_0}), (PK_{i_1}, SK_{i_1}) have been generated by $Gen(1^l)$.

In short, unforgeability means that an efficient adversary cannot generate a valid signature (σ, m, R), simultaneously satisfying 1) $Verify(\sigma, m, R) = 1$, and 2) the honest user does not sign m previously with respect to the same ring R. We use the definition of strong unforgeability (with respect to insider corruption) from [16].

Definition 5. *A ring signature scheme $(Gen, Sign, Verify)$ is unforgeable with respect to insider corruption, if for any PPT adversary $\mathcal{A}$ and for any polynomial $poly(\cdot)$, the probability that $\mathcal{A}$ succeeds in the following game is negligible in $poly(\cdot)$.*

1) Key pairs $\{(PK_i, SK_i)\}_{i=1}^{poly(l)}$ are generated by using $Gen(1^l)$, and the set of public keys $S \overset{def}{=} \{PK_i\}_{i=1}^{poly(l)}$ is given to $\mathcal{A}$.

2) $\mathcal{A}$ is given access to a signing oracle $OSign(\cdot, \cdot, \cdot)$, where $OSign(s, m, R)$ outputs $Sign(SK_s, m, R)$, and we require that $PK_s \in R$, where $1 \leqslant s \leqslant poly(l)$, m is the message waiting to be signed, and $R \subseteq S$ is a ring chosen by $\mathcal{A}$.

3) $\mathcal{A}$ is also given access to a corrupt oracle $Corrupt(\cdot)$, where $Corrupt(i)$ outputs SK_i.

4) $\mathcal{A}$ outputs (σ^*, m^*, R^*), and succeeds if $Verify(\sigma^*, m^*, R^*) = 1$, $\mathcal{A}$ never queried $(*, m^*, R^*)$, and $R^* \subseteq S \setminus C$, where C is the set of corrupted users.

4 Construction of Constant-Size Ring Signature

4.1 System Setup

Let $G = g$ and G_T be groups of prime order p, and $e : G \times G \rightarrow G_T$ be a bilinear map. Let the discrete logarithm problem (DLP) be intractable over G. Let $H_0 : \{0,1\}^* \rightarrow Z_q, H_1 : \{0,1\}^* \rightarrow Z_q$ be cryptographic hash functions. Here, the reason why we use two hash functions is for the simplicity of proof. For $i = 1, ..., n$, each user i has a distinct public key PK_i and a private key x_i such that $PK_i = g^{x_i}$. Let $R = \{PK_1, PK_2, ..., PK_n\}$ be a list of n public keys. For the accumulator presented in Subsection 2.2, we choose y to be the identity function $y(x) = x$; thus in the rest of this paper we simply use the value of function f to represent the accumulated value. Recall that f is defined as $f : (\beta, x) \rightarrow \beta(x + d)$, where $\beta, d \in Z_q$. In addition, we need to run ACC.Gen to generate f and corresponding $P_{pub} = g^d$ as well. The system parameter param $= \{G, G_T, g, p, q, f, P_{pub}, e(g, g), H_0, H_1\}$.

4.2 Signature Generation

Before signing a message, the signer needs to know the message $m \in \{0,1\}^*$, the system parameter param, a list of public keys $R = \{PK_1, PK_2, ..., PK_n\}$, and private key x_s corresponding to the signer's public key PK_s $(1 \leqslant s \leqslant n)$. Let $R' = \{PK_i\}_{i=1, i\neq s}^{n}$, which means that $R = R' \cup \{PK_s\}$. Let $L = \{H_0(PK_i)\}_{i=1}^{n}$ and $L' = \{H_0(PK_i)\}_{i=1, i\neq s}^{n}$. The signature is generated by the following procedures.

1) Compute

$$V = ACC.Eval(f, L),$$
$$W = ACC.Wit(f, L').$$

Note that $V = W(H_0(PK_s) + d)$.

2) Choose randomly $r \leftarrow_R Z_q$, and then compute $U = W + r$ and $P_U = g^U$.

3) Choose randomly $k_1, k_2 \leftarrow_R Z_q$, and compute

$$\Pi = e(g, P_U)^{-k_1} e(P_{pub}, g)^{k_2} e(R', g),$$

where $e(R', g) = \prod_{i=1, i\neq s}^{n} e(PK_i, g)$.

4) Compute $c = H_1(m||V||P_{pub}||P_U||\Pi||R)$.

5) Compute

$$\begin{cases} s_1 = k_1 + cH_0(PK_s), \\ s_2 = k_2 + cr, \\ s_3 = crH_0(PK_s) - x_s. \end{cases}$$

6) The ring signature is

$$\sigma = (c, P_{pub}, P_U, V, s_1, s_2, s_3).$$

Note that the signer mainly performs N pairing operations and three exponentiations.

4.3 Signature Verification

Given a ring signature $\sigma = (c, P_{pub}, P_U, V, s_1, s_2, s_3)$ on a message m and a list of public keys R, the verification algorithm works as follows.

1) Compute $\Pi' = e(g,P_U)^{-s_1}e(P_{pub},g)^{s_2}e(g,g)^{s_3}e(P_{pub},P_U)^{-c}e(g,g^V)^c e(R,g)$, where $e(R,g) = \prod_{i=1}^{n} e(PK_i,g)$.

2) If $c = H_1(m||V||P_{pub}||P_U||\Pi'||R)$, accept the signature; otherwise, reject it.

Theorem 1. *Our constant-size ring signature scheme satisfies completeness.*

Proof. It is easy to see that our ring signature scheme is correct. For a signature $\sigma = (c,V,P_{pub},P_U,s_1,s_2,s_3)$ on a message m and a list of public keys R, we have

$$\begin{cases} P_{pub} = g^d, \\ s_1 = k_1 + cH_0(PK_s), \\ s_2 = k_2 + cr, \\ s_3 = crH_0(PK_s) - x_s, \end{cases}$$

thereby $\Pi' = e(g,P_U)^{-s_1}e(P_{pub},g)^{s_2}e(g,g)^{s_3}e(P_{pub},P_U)^{-c}\ e(g,g^V)^c e(R,g) = e(g,P_U)^{-k_1}e(P_{pub},g)^{k_2}e(R',g)\Delta$, where

$$\begin{aligned}\Delta &= e(g,P_U)^{-cH_0(P_s)}e(g,g)^{crH_0(P_s)}e(P_{pub},g)^{cr} \\ &\quad e(P_{pub},P_U)^{-c}e(g,g^V)^c \\ &= \{e(g,P_U)^{-cH_0(P_s)}e(g,P_U)^{-cd}\}\{e(g,g)^{crH_0(P_s)} \\ &\quad e(g,g)^{crd}\}e(g,g^V)^c \\ &= e(g,P_U)^{-cH_0(P_s)-cd}e(g,g)^{crH_0(P_s)+crd}e(g,g^V)^c \\ &= e(g,g^{U-r})^{-cH_0(P_s)-cd}e(g,g^V)^c \\ &= e(g,g^{W(H_0(P_s)+d)})^{-c}e(g,g^V)^c \\ &= 1.\end{aligned}$$

Thus we can get $\Pi' = \Pi$ and $c = H_1(m||V||P_{pub}||P_U||\Pi'||R)$. □

5 Security Analysis

5.1 Unforgeability

Theorem 2. *Assume that an adversary F has an advantage ε against our constant-size ring signature scheme, asking q_{H_i} queries to random oracles $H_i(i = 0,1)$, and q_s signature queries to signature oracle. Then there is an algorithm B that solves the discrete logarithm problem with probability*

$$\varepsilon' \geqslant \frac{\varepsilon^2}{q_{H_1}} + \frac{(q_{H_1}+q_s)^2}{4^l q_{H_1}} - \frac{\varepsilon(q_{H_1}+q_s)}{2^{l-1}} - \frac{1}{2^l},$$

where l is the security parameter.

Proof. The main idea of the proof is that we use adversary F together with the general forking lemma to generate two forgeries $\sigma = (c,V,P_{pub},P_U,s_1,s_2,s_3)$ and $\sigma' = (c',V',P'_{pub},P'_U,s'_1,s'_2,s'_3)$ satisfying

$$\begin{cases} P_{pub} = g^d, \\ P'_{pub} = g^{d'}, \\ V = W(H_0(PK_s)+d), \\ V' = W'(H_0(PK_s)+d'), \\ U = W + r, \\ U' = W' + r', \\ s_3 = crH_0(PK_s) - x_s, \\ s'_3 = c'r'H_0(PK_s) - x_s, \end{cases} \qquad (2)$$

where c,r,d,W are parameters in the first signature and c',r',d',W' are parameters in the second signature. It holds that $V = V', U = U', P_{pub} = P'_{pub}$ for the same public key $PK_s = g^{x_s}$ and ring R, and the reason why this conditions hold will be described later. In the group of prime order n, we can get $d = d'$ from $P_{pub} = P'_{pub}$; combining with $V = V'$, we can get $W = W'$. Now since we have $U = U'$ and $W = W'$, $r = r'$ would be hold. By subtracting the last two equations with each other in (2), we can get $rH_0(PK_s) = \frac{s_3-s'_3}{c-c'}$ and thus x_s should be

$$x_s = \frac{c(s_3-s'_3)}{(c-c')} - s_3,$$

and once we find the exact x_s, we successfully extract the discrete logarithm of PK_s.

It should be noticed that we assume the same public key is used in both signatures (σ,σ') generated by the adversary in the above statement, but this may not be true in practice. Suppose the adversary uses different public keys as the real signer's public key, it is still possible to solve the discrete logarithm of the signer's public key with non-negligible probability: running the forking lemma algorithm $n+1$ times, where n is the size of the ring. According to the pigeonhole principle, if we got $n+1$ signatures for the same ring with size n, then there are at least one pair of signatures with the same public key, and by (2) we can solve the signer's private key, and DLP is solved. As for the probability with respect to the above event, if we run forking lemma algorithm once and get two different signatures with non-negligible probability, then the probability of running $n+1$ times of the same algorithm should also be non-negligible. Another thing that we should take more care of is that, in the above proof, we assume the accumulator we use in scheme is secure and this may lead to extra assumptions. We are now going to give the actual proof of Theorem 2. Assume there are an adversary F that can attack our scheme with advantage ε, and an algorithm B that tries to solve DLP by using F. To handle the oracle queries, B maintains two lists L_{H_0} and L_{H_1}, which are empty lists at the beginning, and the algorithm B responds to F's oracle queries as follows.

• H_0 *Queries on a Public Key* $pk \in \{0,1\}^*$. If pk has already been defined in L_{H_0}, then algorithm B retrieves ω from L_{H_0}, and returns ω to the adversary F; otherwise B returns a random $\omega \in Z_q$ and adds the tuple (pk, ω) to L_{H_0}.

• H_1 *Queries on a Tuple* $\mu = (m||V||P_{pub}||P_U||\Pi||R)$. If μ has been defined in L_{H_1}, then B retrieves c from L_{H_1} and returns c to F; otherwise B chooses a new random $c \in Z_n$ and adds the tuple (μ, c) to L_{H_1}.

• *Signature Queries on a Tuple* (m, R). First of all, algorithm B randomly picks $\gamma \in \{1, ..., n\}$ where $n = |R|$, and let P_γ be the public key of the real signer. After that, B checks if all public keys in R have been defined in L_{H_0}, and queries H_0 if not. Finally, B calculates the signature as follows.

1) Choose $U \in Z_q$ randomly, and compute $P_U = g^U$.

2) Run *ACC.Gen* to obtain f and P_{pub}, and compute $V = ACC.Eval(f, X)$ where X is a set of results returned by H_0 queries on public keys R.

3) Choose $c, s_1, s_2, s_3 \in Z_q$ randomly, then compute $\Pi = e(g, P_U)^{-s_1} e(P_{pub}, g)^{s_2} e(g, g)^{s_3} e(P_{pub}, P_U)^{-c}$ $e(g, g^V)^c e(R, g)$, and add the new tuple $((m||V||P_{pub}||P_U||\Pi||R), c)$ into L_{H_1} if it is not defined; otherwise, set flag bad = true, and abort.

4) Return $\sigma = (c, V, P_{pub}, P_U, s_1, s_2, s_3)$ as the signature.

Let $Pr[bad]$ denote the probability of the event that flag bad is set to be true, and l denote the security parameter. We bound the accepting probability acc of algorithm B as follows:

$$\begin{aligned} acc &\geqslant \varepsilon - Pr[bad] \\ &\geqslant \varepsilon - \frac{q_{H_1} + q_s}{2^l}. \end{aligned}$$

Now, since we have described how to simulate F's environment, algorithm B can take the input of hash function H_1 as the input and runs the forking lemma algorithm F_B, which with the probability frk returns two forgeries σ and σ' corresponding to the list of public keys R and message m. At this point we have successfully generated the two forgeries, and then we can use the method mentioned at the beginning of the proof to solve the discrete logarithm problem with the probability

$$\begin{aligned} \varepsilon' &\geqslant frk \\ &\geqslant \frac{acc^2}{q_{H_1}} - \frac{1}{2^l} \\ &= \frac{\varepsilon^2}{q_{H_1}} + \frac{(q_{H_1} + q_s)^2}{4^l q_{H_1}} - \frac{\varepsilon(q_{H_1} + q_s)}{2^{l-1}} - \frac{1}{2^l}. \end{aligned}$$

We still have to show why the equalities in (2) in both executions of B hold. In the case that F returned (σ, σ'), let J be the index that algorithm B returned after both executions by F_B. In the first execution of B, $h_J = c$ is assigned to $L_{H_1}[m||V||P_{pub}||P_U||\Pi||R]$ when F makes its first query $(m||V||P_{pub}||P_U||\Pi||R)$. Similarly, in B's second execution, $h'_J = c'$ is assigned to $L_{H_1}[m'||V'||P'_{pub}||P'_U||\Pi'||R']$ when F queries $(m'||V'||P'_{pub}||P'_U||\Pi'||R')$. Up to this point, the environments of F provided by B are the same in both the first and the second execution, because B uses the same inputs, random tape and values $h_1, ..., h_{J-1}$ to generate $F's$ input, random tape and oracle responses. Thus the two executions of F are identical up to this point, and the arguments of both hash queries must be the same. It implies that $V = V'$, $U = U'$, and $P_{pub} = P'_{pub}$. □

5.2 Anonymity

In order to prove that our scheme satisfies anonymity, we prove that our scheme is perfect zero knowledge. In zero knowledge, the prover knows some secret knowledge and needs to convince the verifier about the secret without leaking any other information about the secret knowledge. It means that there exists a simulator that can simulate the response of the prover, and the verifier cannot distinguish between the real prover and the simulator. For more details about the proof of the zero knowledge, readers may refer to [26]. In our scheme, the simulator chooses randomly $P_{pub}, U, c, s_1, s_2, s_3 \in Z_q$, computes V, P_U, and then computes $\Pi = e(g, P_U)^{-s_1} e(P_{pub}, g)^{s_2} e(g, g)^{s_3} e(P_{pub}, P_U)^{-c} e(g, g^V)^c e(R, g)$. We can see that the distribution of the simulation is the same as the real transcript, which completes the proof.

6 Comparison

In this section, we make a comparison among some of the existing ring signatures on their efficiency, and the efficiency is evaluated by the number of different operations. Details can be found in Table 2. The results are calculated on the assumption that the size of

Table 2. Efficiency Comparison Among Some Existing Ring Signatures

Reference	Size	Number of Operations Used					
		EXP	$HASH$	XOR	ADD	$MULTI$	$PAIRING$
[5]	$O(n)$	n	1	$n-1$	$\perp$	$\perp$	$\perp$
[13]	$O(n)$	n	n	$\perp$	n	$\perp$	$\perp$
[15]	$O(n)$	$2n$	1	$\perp$	$\perp$	n	$\perp$
[14]	$O(n)$	$4n-1$	n	$\perp$	1	1	$\perp$
[16]	$O(n)$	–	–	–	–	–	–
[17]	$O(\sqrt{n})$	$12\sqrt{n}+5$	$\perp$	$\perp$	$2\sqrt{n}+7$	$n+5\sqrt{n}+9$	$\perp$
[18]	$O(\sqrt{n})$	$11\sqrt{n}+1$	1	$\perp$	$2\sqrt{n}+10$	$4\sqrt{n}+11$	$\perp$
[19]	$O(\sqrt{n})$	$12\sqrt{n}+4$	$\perp$	$\perp$	$6\sqrt{n}+5$	$n+6\sqrt{n}+10$	$\perp$
[11]	$O(1)$	–	–	–	–	–	–
[21]	$O(1)$	$2n+1$	$\perp$	$\perp$	2	$1+n^2$	1
Ours	$O(1)$	2	$n+1$	$\perp$	$2n+4$	$3n+4$	$n-1$

the ring is n. And only the signature generation process is taken into considered.

To describe the number of operations used in each scheme, we denote by EXP the number of exponential operations, by $HASH$ the number of hash operations, by XOR the number of exclusive-or operations, by ADD the number of modular addition or subtraction operations, by $MULTI$ the number of modular multiplication operations, and by $PAIRING$ the number of bilinear pairing operations. And $\perp$ means that there is no such operation in corresponding signature schemes.

Besides, for some schemes such as [16] and [11], we use No Concrete Method (–) because the original paper does not give the concrete algorithm to generate the signature or these schemes use some other signature schemes or public key encryption without mentioning a specific name or process, therefore the overall efficiency cannot be estimated.

The first four schemes[5,13-15] in Table 2 are size of $O(n)$, although they cost less on almost all the operations. In scheme [16], it uses both public-key encryption and standard signature scheme, but these two schemes both have different efficiencies in different implementations; thus the overall efficiency cannot be calculated exactly.

The next three schemes[17-19] are all of size $O(\sqrt{n})$, and they use similar ideas to prove the set membership, thereby their efficiencies are quite similar. Besides, for schemes [18] and [19], additional one-time signature[27-28] calculations need to be added.

The only two schemes of constant size were proposed by Dodis *et al.*[11] and Bose *et al.*[21] respectively, but the concrete methods are not given in scheme [11]. And scheme [21] also needs to add five $GSProve(\cdot)$ calculations, but the concrete steps of $GSProve(\cdot)$ are not included in the original paper[21], and this process costs much. Finally, our scheme not only does not rely on other signature schemes but also is quite easy to implement for it only contains few steps. And most importantly, it is also proved to be secure.

7 Conclusions

In this paper, we introduced a new constant-size ring signature scheme which is more practical than existing schemes, and a proof of security was given in details at the same time. Additionally, we made a detail comparison between some of the existing ring signatures on the efficiency of signature generation, including schemes of linear size, sub-linear size, and constant size. All the schemes have both advantages and disadvantages. From Table 2, we can see that in each size range (e.g., linear, sub-linear), the efficiency of each scheme has not much difference, but some of their implementations are based on some other complicated schemes (e.g., Full Boneh-Boyen signature in [21]), which makes them not practical.

There are still many interesting problems to be solved. For example, it would be valuable to explore the possibility of achieving a constant-size ring signature with shorter length than our scheme, or with higher efficiency in implementation. Using our scheme to construct a real application such as e-voting system or cryptocurrency is also an interesting problem.

References

[1] Satoshi N. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf, Mar. 2018.

[2] Möser M. Anonymity of bitcoin transactions. In *Proc. Münster Bitcoin Conf.*, July 2013, pp.17-18.

[3] Ron D, Shamir A. Quantitative analysis of the full bitcoin transaction graph. In *Proc. the 17th Int. Conf. Financial Cryptography and Data Security*, April 2013, pp.6-24.

[4] Androulaki E, Karame G O, Roeschlin M, Scherer T, Capkun S. Evaluating user privacy in bitcoin. In *Proc. the 17th Int. Conf. Financial Cryptography and Data Security*, April 2013, pp.34-51.

[5] Rivest R L, Shamir A, Tauman Y. How to leak a secret: Theory and applications of ring signatures. In *Theoretical Computer Science*, Goldreich O, Rosenberg A L, Selman A L (eds.), Springer, 2006, pp.164-186.

[6] Chaum D, van Heyst E. Group signatures. In *Proc. Workshop on the Theory and Appl. Cryptographic Techniques*, April 1991, pp.257-265.

[7] van Saberhagen N. Cryptonote v 2.0, 2013. https://cryptonote.org/whitepaper.pdf, March 2018.

[8] Noether S, Mackenzie A, The Monero Research Lab. Ring confidential transactions. *Ledger*, 2016, 1: 1-18.

[9] Jakobsson M, Sako K, Impagliazzo R. Designated verifier proofs and their applications. In *Proc. Int. Conf. Theory and Appl. Cryptographic Techniques*, May 1996, pp.143-154.

[10] Chow S S M, Liu J K, Wong D S. Robust receipt-free election system with ballot secrecy and verifiability. In *Proc. Network and Distributed System Security Symp.*, Feb. 2008.

[11] Dodis Y, Kiayias A, Nicolosi A, Shoup V. Anonymous identification in ad hoc groups. In *Proc. Int. Conf. Theory and Appl. Cryptographic Techniques*, May 2004, pp.609-626.

[12] Naor M. Deniable ring authentication. In *Proc. the 22nd Annu. Int. Cryptology Conf.*, August 2002, pp.481-498.

[13] Abe M, Ohkubo M, Suzuki K. 1-out-of-n signatures from a variety of keys. In *Proc. the 8th Int. Conf. Theory and Appl. Cryptology and Information Security*, December 2002, pp.415-432.

[14] Liu J K, Wei V K, Wong D S. Linkable spontaneous anonymous group signature for ad hoc groups. In *Proc. the 9th Australasian Conf. Information Security and Privacy*, July 2004, pp.325-335.

[15] Boneh D, Gentry C, Lynn B, Shacham H. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc. Int. Conf. Theory and Appl. Cryptographic Techniques*, May 2003, pp.416-432.

[16] Bender A, Katz J, Morselli R. Ring signatures: Stronger definitions, and constructions without random oracles. In *Proc. the 3rd Theory of Cryptography Conf.*, March 2006, pp.60-79.

[17] Chandran N, Groth J, Sahai A. Ring signatures of sub-linear size without random oracles. In *Proc. the 34th Int. Colloquium on Automata Languages and Programming*, July 2007, pp.423-434.

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

[19] Ghadafi E M. Sub-linear blind ring signatures without random oracles. In *Proc. the 14th IMA Int. Conf. Cryptography and Coding*, December 2013, pp.304-323.

[20] Fuchsbauer G. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. IACR Cryptology ePrint Archive Report 2009/320, 2009. https://eprint.iacr.org/2009/320.pdf, Mar. 2018.

[21] Bose P, Das D, Rangan C P. Constant size ring signature without random oracle. In *Proc. the 20th Australasian Conf. Information Security and Privacy*, July 2015, pp.230-247.

[22] Galbraith S D, Paterson K G, Smart N P. Pairings for cryptographers. *Discrete Applied Mathematics*, 2008, 156(16): 3113-3121.

[23] Nguyen L. Accumulators from bilinear pairings and applications. In *Proc. Cryptographers' Track at the RSA Conference*, February 2005, pp.275-292.

[24] Ben-Sasson E, Chiesa A, Garman C, Green M, Miers I, Tromer E, Virza M. Zerocash: Decentralized anonymous payments from bitcoin. In *Proc. IEEE Symp. Security and Privacy* (*SP*), May 2014, pp.459-474.

[25] Bellare M, Neven G. Multi-signatures in the plain public-key model and a general forking lemma. In *Proc. the 13th ACM Conf. Computer and Communications Security*, October 30-November 3, 2006, pp.390-399.

[26] Goldwasser S, Micali S, Rackoff C. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 1989, 18(1): 186-208.

[27] Bleichenbacher D, Maurer U. On the efficiency of one-time digital signatures. In *Proc. Int. Conf. Theory and Appl. Cryptology and Information Security*, November 1996, pp.145-158.

[28] Perrig A. The BiBa one-time signature and broadcast authentication protocol. In *Proc. the 8th ACM Conf. Computer and Communications Security*, November 2001, pp.28-37.

Meng-Jun Qin received his B.S. degree in computer science from Donghua University, Shanghai, in 2016. He is currently pursuing his M.S. degree in computer science in Fudan University, Shanghai. His research interests include blockchain, consensus algorithm and ring signature.

Yun-Lei Zhao received his Ph.D. degree in computer science from Fudan University, Shanghai, in 2004. At the same year he joined Hewlett-Packard European Research Center, Bristol, U.K., as a post-doctoral researcher. Since 2005, he has been with Fudan University, and is currently a professor with the School of Computer Science, Fudan University, Shanghai. His research interests are the theory and applications of cryptography.

Zhou-Jun Ma received his M.S. degree in computer science from Fudan University, Shanghai, in 2017. He is currently working at Microsoft in Vancouver. His research interests mainly focus on digital signatures.