EFFICIENT TRACEABLE RING SIGNATURE SCHEME WITHOUT PAIRINGS

KE GU*, XINYING DONG AND LINYU WANG
School of Computer and Communication Engineering
Changsha University of Science and Technology
Changsha 410114, China

(Communicated by Sihem Mesnager)

Abstract. Although currently several traceable (or linkable) ring signature schemes have been proposed, most of them are constructed on pairings. In this paper, we present an efficient traceable ring signature (TRS) scheme without pairings, which is based on the modified EDL signature (first proposed by D.Chaum et al. in Crypto 92). Compared with other ring signature schemes, the proposed scheme does not employ pairing computation and has some computational advantages, whose security can be reduced to the computational Diffie-Hellman (CDH) and decisional Diffie-Hellman (DDH) assumptions in the random oracle model. Also, the proposed scheme is similar to certificate-less signature scheme, where user and key generating center make interaction to generate ring key. We give a formal security model for ring signature and prove that the proposed scheme has the properties of traceability and anonymity.

1. Introduction

1.1. Background. Ring signature [1, 10, 21, 23, 45, 51, 54] allows ring member to hide his identifying information to a ring when ring member signs any message, thus ring signature only reveals the fact that a message was signed by possible one of ring members (a list of possible signers). Ring signature is also called as a special group signature [18]. However, compared with group signature, ring signature has more advantages: the group (ring) must not be constructed by a group manager, who can revoke the anonymity of any signer or identify the real group signer; additionally, because a list of possible signers must be constructed to form a group, some intricate problems need to be solved in a group signature scheme, such as joining the new members and the revocation of group members. Although ring signature can provide more flexibility and full anonymity, it is vulnerable to keep the signers from abusing their signing rights. Namely, it is infeasible for the verifier to determine whether the signatures are generated by the same signer on the same event. Thus, in a practical ring signature scheme, the third trusted authority or the verifier must be able to know who signs the messages on the same event many times and the verifier can not accept the signatures generated by the same signer on the same event [2, 8, 13, 35, 39, 42].

2010 Mathematics Subject Classification: Primary: 58F15, 58F17; Secondary: 53C35.
Key words and phrases: Ring signature, traceability, pairings, security model.

This work is supported by the National Natural Science Foundations of China (No.61402055), the Hunan Provincial Natural Science Foundation of China (No.2018JJ2445) and the Open Research Fund of Key Laboratory of Network Crime Investigation of Hunan Provincial Colleges (No.2017WLFZZC003).

* Corresponding author: Ke Gu.

©2020 AIMS
Traceable ring signature\textsuperscript{1} [26] is a ring signature that restricts abusing anonymity. Unlike group signature has too strong a traceability characteristic and ring signature has too strong an anonymity characteristic, traceable ring signature has the balance characteristic of anonymity and traceability. Traceable ring signature provides restricted anonymity and traceability. In a traceable ring signature scheme, traceable ring signature can provide full anonymity for the responsible or honest signer when the signer signs any message, and provide traceability for the verifier (or the third trusted authority) to determine whether the signatures are generated by the same signer on the same event when the irresponsible signer abuses anonymity in some applications. In order to achieve this requirement of traceable ring signature, we need to consider the two notions “one-more unforgeability” and “double-spending traceability” [15, 16, 26] in the context of ring signature, which originate from blind signature. First, any user cannot generate a “one-more” new signature after he obtained a signature from the original signer. Second, if an irresponsible user signs any message twice on the same event, the signatures generated by the user can be traced to reveal the identity of the signer [12, 43]. In the second notion, a responsible user can be anonymously protected. Obviously, traceable ring signature can provide more practicality because of its restricted anonymity in many no full anonymous applications.

Currently ring signatures are used in many different applications, such as whistle blowing [45], anonymous authentication for ad-hoc network [39], e-voting [20] and e-cash [48], non-interactive deniable authentication [47] and multi designated verifiers signature [36], etc. Because ring signature is not linkable, no one can determine whether two ring signatures are generated by the same signer. Thus, it exists high risk that ring signatures are used in e-voting and e-cash. For example, a user is only allowed to sign a voting message once for anonymous e-voting, thus when a malicious user signs a message twice for double votes in anonymous e-voting, no one can find the two signatures are linkable so as to detect the irregularity of the malicious user. Obviously, traceable ring signature is suitable for the kind of applications, because it can find the two signatures are linkable. There also are other applications for traceable ring signature. In the “off-line” anonymous e-cash systems, a user is permitted to anonymously sign a message once during one cash transaction, thus traceable ring signature is a natural choice for this application [26]. Damgard et al. [22] proposed an unclonable group identification without the group manager, traceable ring signature is also suitable for this application because of not employing the group manager and its balance of anonymity and traceability.

\textbf{EDL signature}

The EDL signature [17] and its variant [33] are respectively proposed in 1992 and 1999. Because the computations of the EDL signature do not employ pairings, the efficiency of the schemes is very high. In 2003, Goh et al. [27] proved the security of the EDL signature may be reduced to the CDH assumption in the random oracle model. In 2005, Chevallier-Mames [19] further improved the efficiency of the EDL signature by offline/online computation and signature coupon [46], whose security may also be reduced to the CDH assumption in the random oracle model.

1.2. Our contributions. In this paper, we present a public key-based traceable ring signature scheme without pairings. Also, we give the formal security models for traceable ring signature. Under our security models, the proposed scheme is

\textsuperscript{1}This notion is closely related to linkable ring signature in [4, 39, 40, 41].
proved to have the properties of anonymity and traceability with enough security in the random oracle model. In this paper, our contributions are as follows:

- We present a public key-based traceable ring signature scheme without pairings, which is based on the modified EDL signature. By modifying the EDL signature from [19, 27], we twice use the modified EDL signature to build a complete traceable ring signature scheme: a) we first use the modified EDL signature to construct the partial member private (ring) keys when the users join a ring; b) we again use the modified EDL signature to generate the valid ring signatures. Also, the proposed scheme is similar to certificateless signature scheme, where user and key generating center make interaction to generate ring key.

- We present a framework for TRS and show a detailed security model for TRS. Compared with the security models of TRS [4, 26], we integrate the Fujisaki et al.'s frame and the Au et al.'s frame to our security model. In our security model, we consider four situations for the security of TRS and further strengthen our security model on public key cryptography, where we still consider the trusted authority is partially trusted. Under our security model, the proposed TRS scheme is proved to be secure and has a security reduction to the simple standard assumptions (computational Diffie-Hellman assumption and decisional Diffie-Hellman assumption) in the random oracle model. So, no poly-time adversary can produce a valid traceable ring signature on any messages when the adversary may adaptively be permitted to choose messages after executing oracles.

- Compared with other traceable (or linkable) ring signature schemes proposed by [4, 25, 26, 40, 55], the proposed TRS scheme does not employ pairing computation, and has some computational advantages (the comparisons of the schemes are given in Section 7).

1.3. Outline. The rest of this paper is organized as follows. In Section 2, we discuss the related works about TRS. In Section 3, we review the complexity assumptions on which we build. In Section 4, we show a framework for TRS. In Section 5, we set up the security model for TRS. In Section 6, we propose a traceable ring signature scheme without pairings under our framework for TRS. In Section 7, we analyze the efficiency and security of the proposed scheme. Finally, we draw our conclusions in Section 8.

2. Related work

Liu et al. [39] first proposed the notion of linkable ring signature. In their scheme, if an irresponsible user anonymously signs any message twice on the same event, the two signatures generated by the user can be linked. Base on this notion, some similar schemes were proposed in [3, 39, 40, 41, 48, 49]. In [39, 40], the proposed schemes cannot resist the attack that an irresponsible signer forges the signature of a honest signer so as to make the honest signer accused of “double-signing”. In [3, 49], the proposed schemes overcome this weakness, but the security conditions are more complicated. In [48], Tsang et al. proposed a short linkable ring signature scheme, which is based on the group identification scheme from [23]. Their scheme provides weak traceability, namely it can only detect the linkable ring signatures. In [49], Tsang et al. proposed a separable linkable threshold ring signature scheme.
where the threshold setting is to restrict abusing signing. However, their scheme is complicated. In [38], Liu et al. proposed a revocable ring signature scheme, which supports that any ring member may revoke the anonymity of the real signer when the ring signature is proved to be argumentative. Their scheme provides that all the ring members can reveal the identity of the real signer of any ring signature generated on behalf of their ring.

In 2007 and 2011, Fujisaki et al. [25, 26] proposed two traceable ring signature schemes based on public key cryptography, and a security model of traceable ring signature was formally proposed. In their scheme, if two signatures are linked, the identity of this signer will be revealed. In other words, the anonymity of the signer will be revoked if and only if the signer generates two ring signatures on the same event. Compared with revocable ring signature [38], traceable ring signature needs the condition of revoking anonymity that the same signer generates two ring signatures on the same event. Although the scheme proposed by [25] is constructed without random oracles, the signature size of the scheme is sub-linear with \(O(\sqrt{n})\), where \(n\) is the number of users in the ring. In 2012, Zeng et al. [53] proposed a conditionally anonymous ring signature scheme, where one user can confirm his signature by himself through a confirmation protocol when he signed a signature. In 2013, Yuen et al. [52] proposed an efficient linkable and/or threshold ring signature scheme without random oracles. However, the signature size of their scheme is still sub-linear with \(O(d \cdot \sqrt{n})\), where \(d\) is the threshold value and \(n\) is the number of users in the ring. In 2014, Liu et al. [37] proposed a linkable ring signature scheme with unconditional anonymity. However, the signature size of their scheme is still linear with \(O(n)\).

With the rapid development of identity-based cryptography [9, 11, 28, 29, 30, 44, 50], many researchers proposed many identity-based signature (IBS) schemes in the random oracle model or standard model [7, 14, 31, 44]. Also, with these identity-based signature schemes, a lot of variants, such as the identity-based ring signature schemes [4, 5, 6, 54], the identity-based group signature schemes [24, 32], etc., have also been proposed. In 2006, Au et al. [5] proposed a constant size identity-based linkable and revocable-iff-linked ring signature scheme. However, their scheme was later proved to be insecure [34]. In 2012, Au et al. [4] proposed a new identity-based event-oriented linkable ring signature scheme with an option as revocable-iff-linked. With this option, if a user generates two linkable ring signatures in the same event, everyone can compute his identity from these two signatures. However, their scheme needs to employ pairing computation.

3. Preliminaries

Definition 3.1. Computational Diffie-Hellman (CDH) Problem: Let \(G_1\) be a group of prime order \(q\) and \(g\) be a generator of \(G_1\); for all \((g, g^a, g^b) \in G_1\), with \(a, b \in \mathbb{Z}_q\), the CDH problem is to compute \(g^{a \cdot b}\).

Definition 3.2. The \((h, \varepsilon)\)-CDH assumption holds if no \(h\)-time algorithm can solve the CDH problem with probability at least \(\varepsilon\).

Definition 3.3. Decisional Diffie-Hellman (DDH) Problem: Let \(G_1\) be a group of prime order \(q\) and \(g\) be a generator of \(G_1\); for all \((g, g^a, g^b, g^c) \in G_1\), with \(a, b, c \in \mathbb{Z}_q\), the DDH problem is to decide \(a \cdot b \equiv c\).

Definition 3.4. The \((h, \varepsilon)\)-DDH assumption holds if no \(h\)-time algorithm can solve the DDH problem with probability at least \(\varepsilon\).
4. A FRAMEWORK FOR TRS

In the section, we present a framework for TRS. In [26], Fujisaki et al. proposed a syntax for TRS. In their definition, any verifier who obtains two signatures generated by the same signer can disclose the identity of the signer through tracing verification as long as the traceable condition is satisfied. Therefore, in order to better satisfy the anonymity of ring signatures, in our definition we consider to narrow the scope of entities that can use this tracing function. We add the trusted authority to make tracing verification (that becomes a centralized tracing method), where only the trusted authority can perform the function of tracing signatures and other verifiers cannot make tracing verification. However, we consider if the trusted authority alone generates a ring key for each ring member, then when the trusted authority is corroded it can arbitrarily frame any attacked members. Therefore, in our definition we set a protocol between the trusted authority and the ring members, where the trusted authority only generates a user’s partial ring key for each ring member and then each member generates the final ring key by self. Such construction makes the trusted authority not frame an honest signer when some signatures need to be traced.

Definition 4.1. Traceable Ring Signature Scheme: Let $\text{TRS}=(\text{System-Setup}, \text{Generate-Key}, \text{Generate-Ring Key}, \text{Sign}, \text{Verify}, \text{Trace-User})$ be a traceable ring signature scheme. In $\text{TRS}$, all algorithms are described as follows:

1): System-Setup: The randomized algorithm run by the trusted authority inputs a security parameter $1^k$. Then the algorithm outputs all system parameters $TRK$ and a master key $spk$ on the security parameter $1^k$, and publishes a ring public key $pk_R$.

2): Generate-Key: The randomized algorithm run by a ring member generates his public/private key pair $(pk_i, sk_i)$ with $i \in \{1, 2, ..., n\}$, where $n$ is the maximal number of users in a ring, $pk_i$ is the public key of the ring member $i$ and $sk_i$ is the private key of the ring member $i$.

3): Generate-Ring Key: The randomized algorithm run by a ring member inputs $TRK$ and $pk_R$, and then the following steps are finished:
   • The algorithm run by the trusted authority outputs a user’s partial ring key $psk_i$ to the ring member according to the master key $spk$, where $i \in \{1, 2, ..., n\}$ (where $n$ is the maximal number of users in the ring).
   • The algorithm run by the ring member outputs the ring key $csk_i$ according to $psk_i$ and $sk_i$, where $i \in \{1, 2, ..., n\}$.

4): Sign: The randomized algorithm is a standard traceable ring signature algorithm. A ring member with the ring key $csk_i$ needs to sign a message $M \in \{0, 1\}^*$ on an event identifier $4 \in \{0, 1\}^*$. The algorithm run by the ring member inputs $(TRK, pk_R, csk_i, RL_PK, M, E)$, and then outputs a signature $\sigma$, where $RL_PK$ is a public key list including all public keys of the ring members belong to this ring and $\sigma \in \{0, 1\}^* \cup \{\bot\}$.

5): Verify: The signature verifiers verify a standard traceable ring signature $\sigma$. The deterministic algorithm run by a signature verifier inputs $(TRK, pk_R, RL_PK, M, E, \sigma)$, and then outputs the boolean value, accept or reject.

---

We assume that the trusted authority cannot modify the outputs of the tracing algorithm.

In this paper, the event identifier is only the name (tag or index) of an event, such as the name of vote or the index of cash transaction.
6): Trace-User: The trusted authority traces a real ring member (signer) by two traceable ring signatures $\sigma_1$ on $M_1$ and $\sigma_2$ on $M_2$. The deterministic algorithm run by the trusted authority inputs $(TRK, RL_PK, \{M_1, \sigma_1\}, \{M_2, \sigma_2\}, E)$, and then outputs one of the following results: “the public key $pk$ of the real signer”, or “Independent” or “Linked”\textsuperscript{5}, where $pk \in RL_PK$.

The correctness of TRS requires that for any $(TRK, pk_R, spk) \leftarrow System-Setup(1^k)$, $(sk_i, pk_i) \leftarrow Generate-key(\star)$ for all $i \in \{1, 2, \ldots, n\}$, $csk_i \leftarrow Generate-Ring Key(TRK, pk_R)$ for all $i \in \{1, 2, \ldots, n\}$, $M \in \{0, 1\}^*$ and $E \in \{0, 1\}^*$, then:

$$\Pr[Verify(TRK, pk_R, RL_PK, M, E, Sign(TRK, pk_R, csk_i, RL_PK, M, E))=1]=1.$$ 

The traceability of TRS requires that for any $(TRK, pk_R, spk) \leftarrow System-Setup(1^k)$, $(sk_i, pk_i) \leftarrow Generate-key(\star)$ for all $i \in \{1, 2, \ldots, n\}$, $csk_i \leftarrow Generate-Ring Key(TRK, pk_R)$ for all $i \in \{1, 2, \ldots, n\}$, $M_1, M_2 \in \{0, 1\}^*$ and $E \in \{0, 1\}^*$, if $\sigma_1 \leftarrow Sign(TRK, pk_R, csk_i, RL_PK, M_1, E)$ with $i \in \{1, 2, \ldots, n\}$ and $\sigma_2 \leftarrow Sign(TRK, pk_R, csk_j, RL_PK, M_2, E)$ with $j \in \{1, 2, \ldots, n\}$ and $\sigma_1 \neq \sigma_2$:

\[
\text{Trace-User}(TRK, RL_PK, \{M_1, \sigma_1\}, \{M_2, \sigma_2\}, E) = \begin{cases} 
\text{"Independent"}, & \text{if } i \neq j \\
\text{"Linked"}, & \text{else if } M_1 = M_2 \\
\text{"the public key } pk \text{ of the real signer"}, & \text{otherwise}
\end{cases}
\]

5. Security model

In a secure TRS scheme, we need to consider the two notions “one-more unforgeability” and “double-spending traceability”. First, any user cannot forge a new signature. Second, the anonymity of the signer will be revoked if and only if the signer generates two ring signatures on the same event. Thus, we consider that a fully secure TRS scheme must meet the following security requirements according to [4, 26]:

- **Unforgeability**: A valid TRS signature must be signed by a valid ring member (signer). Therefore, no poly-time adversary can produce a valid TRS signature on any messages when the adversary may adaptively be permitted to choose messages after executing signature oracle. Then we split the requirement to the following two small security notions\textsuperscript{6}:

  a): the first one is called security against linkability attacks\textsuperscript{7}, which requires that every two signatures generated by the same signer with respect to the same tag of event are linked. Therefore, the total number of signatures with respect to the same tag cannot exceed the total number of ring members in the tag if every any two signatures are not linked.

\textsuperscript{5} In [26], Fujisaki et al. consider the result “Linked” does not mean that two same signatures are always generated by the same signer because anyone can make a “dead” copy of any signature. So, they further divide the outputs of the tracing algorithm to three results. The result “Linked” is not equal to the concept of linkability (or traceability). In this paper, we also consider a “dead” copy of any signature is easy to confuse the outputs of the tracing algorithm.

\textsuperscript{6} The two security notions should be more detailedly expanded from the correctness of unforgeability.

\textsuperscript{7} In this paper, we also call the notion **Traceability**: anyone who creates two signatures for different messages with respect to the same tag of event can be traced, where the trace can be done only with pairs of message/signature pairs and tag.
b): the second one is called security against *exculpability attacks*, which
requires that an honest ring member cannot be accused of signing twice
with respect to the same tag. Therefore, an adversary cannot produce a
traceable ring signature along with one signature generated by a target
member (an attacked member) such that the signatures can designate
the target member in the presence of the public traceable method. This
forgery is also infeasible even after the adversary has corrupted all ring
members but the target member.

- **Anonymity**: As long as a signer does not sign two different messages with
  respect to the same tag, the identity of the signer is indistinguishable from
  any of the possible ring members. In addition, any two signatures generated
  with respect to two distinct tags are always unlinkable. So, it is infeasible for
  anyone to determine whether they are generated by the same signer.

Then, based on the above situations, we propose a complete security model for
traceable ring signature. To make our security model easier to understand, we
construct several algorithms interacting with adversary, which may make attack
experiments to the traceable ring signature schemes in the above situations. In our
security model, we maximize adversary’s advantage, and assume that all attacking
conditions needed by adversary hold and adversary may forge signatures after lim-
itedly querying oracles. Additionally, as we add a protocol to generate the final
ring key between the trusted authority and the ring members, we need to consider
the security of traceable ring signature when the trusted authority is untrusted (or
partially trusted). So, to maximize adversary’s advantage, the adversary can query
the oracle for the partial ring key.

In our security model, we assume there are \( n + 1 \) users in a traceable ring signature
scheme (\( n \in \mathbb{N} \) is a maximal number of ring members in a ring), and at least one
user \( u^* \) of \( n + 1 \) users is not corrupted by adversary. Because we consider the trusted
authority is partially trusted in the unforgeability, we add the corruption ability to
the adversary in Definition 5.1, where the adversary can query the system for the
partial ring key.

**Definition 5.1.** Unforgeability of A Traceable Ring Signature Scheme: Let \( TRS=(\)
*System-Setup, Generate-Key, Generate-Ring Key, Sign, Verify, Trace-
User*) be a traceable ring signature scheme. Additionally, we set that \( k \) is a secure
parameter, and \( \Pr(B_{U,TRS}(k,A)=1) \) is the probability that the algorithm \( B_{U,TRS} \)
returns 1. Then the advantage that the adversary \( A \) breaks \( TRS \) is defined as
follows:

\[
\text{Adv}_{TRS}^{u,TRRS} = \Pr(B_{U,TRS}(k,A)=1),
\]

where \( q_r \) is the maximal number of “Register-User Key” oracle queries, \( q_p \) is the
maximal number of “Generate-Partial Ring Key” oracle queries, \( q_s \) is the maximal
number of “Sign” oracle queries and \( h \) is the running time of \( B \). If the advantage
that the adversary breaks \( TRS \) is negligible, then the scheme \( TRS \) is secure.

According to the Definition 5.1, the algorithm \( B_{U,TRS} \) is described as follows:

1. **Setup**: Running *System-Setup*, \( (TRK,pk_R,spk) \leftarrow \text{System-Setup}(1^k) \), and
then \( TRK \) and \( pk_R \) are passed to \( A \).
2. **Queries**: \( A \) makes queries to the following oracles for polynomially many times:
   - **Register-User Key()**: Given the public parameters \( TRK \), the oracle out-
     puts the private/public key pair \( (sk,pk) \) of a user to \( A \).
• **Generate-Partial Ring Key()**: Given the public parameters TRK and the ring public key \( pk_R \), the oracle outputs the partial ring key \( psk \).

• **Sign()**: Given the public parameters TRK, the ring public key \( pk_R \), the public key list \( RL_{PK} \), the message \( \mathcal{M} \) and the event identifier \( \mathcal{E} \), the oracle returns a signature \( \sigma \) to \( A \), where \( \sigma \in \{0,1\}^* \cup \{ \perp \} \).

(3) **Forgery**: \( A \) outputs its forgery, \((\mathcal{M}^*, \mathcal{E}^*, \sigma^*)\) for \( RL_{PK}^* \), where \( RL_{PK}^* \) is a public key list including all public keys of the ring members belong to this ring. It succeeds if

(a): \( 1 \leftarrow \text{Verify}(TRK, pk_R, RL_{PK}^*, \mathcal{M}^*, \mathcal{E}^*, \sigma^*) \);

(b): \( A \) did not query \( \text{Sign} \) on inputs \( RL_{PK}^*, \mathcal{M}^* \) and \( \mathcal{E}^* \).

The following two definitions are expanded from unforgeability (see [4, 26] for more details).

An adversary cannot generate two signatures in the same event without being linked. We generalize the notion that an adversary with \( t \) user (ring member) private keys cannot create \( t + 1 \) signatures in the same event without being linked, where the notion comes from the work of [4]. We consider that even if the adversary can have the maximal ability to control the privates keys of \( t \) users, it cannot create \( t + 1 \) signatures in the same event. Similarly, we add the corruption ability to the adversary in Definition 5.2, where the adversary can query the system for the partial ring key.

**Definition 5.2.** Linkability (or Traceability) of A Traceable Ring Signature Scheme: Let \( TRS = (\text{System-Setup, Generate-Key, Generate-Ring Key, Sign, Verify, Trace-User}) \) be a traceable ring signature scheme. Additionally, we set that \( k \) is a secure parameter, and \( \text{Pr}(B_{L,TRS}(k,A)=1) \) is the probability that the algorithm \( B_{L,TRS} \) returns 1. Then the advantage that the adversary \( A \) breaks \( TRS \) is defined as follows:

\[
\text{Adv}_{TRS}^{l_{trs-a}}(k, q_r, q_p, q_s, t, h) = \text{Pr}(B_{L,TRS}(k,A)=1),
\]

where \( q_r \) is the maximal number of “Register-User Key” oracle queries, \( q_p \) is the maximal number of “Generate-Partial Ring Key” oracle queries, \( q_s \) is the maximal number of “Sign” oracle queries, \( t \) is the number of user (ring member) private keys possessed by \( A \) and \( h \) is the running time of \( B \). If the advantage that the adversary breaks \( TRS \) is negligible, then the scheme \( TRS \) is secure.

According to the Definition 5.2, the algorithm \( B_{L,TRS} \) is described as follows:

1) **Setup:** Running System-Setup, \((TRK, pk_R, spk) \leftarrow \text{System-Setup}(1^k)\), and then \( TRK \) and \( pk_R \) are passed to \( A \).

2) **Queries:** \( A \) makes queries to the following oracles for polynomially many times:
   • **Register-User Key()**: Given the public parameters \( TRK \), the oracle outputs the private/public key pair \((sk, pk)\) of a user to \( A \).
   • **Generate-Partial Ring Key()**: Given the public parameters \( TRK \) and the ring public key \( pk_R \), the oracle outputs the partial ring key \( psk \).
   • **Sign()**: Given the public parameters \( TRK \), the public key list \( RL_{PK} \), the message \( \mathcal{M} \) and the event identifier \( \mathcal{E} \), the oracle returns a signature \( \sigma \) with respect to a ring public key to \( A \), where \( \sigma \in \{0,1\}^* \cup \{ \perp \} \).

3) **Forgery**: \( A \) outputs its forgery, a set of tuples \((\mathcal{M}_i^*, \mathcal{E}_i^*, \sigma_i^*; RL_{PK_i}^*)\) with \( i \in \{1,...,t+1\} \). It succeeds if
   (a): \( 1 \leftarrow \text{Verify}(TRK, pk_R^*, RL_{PK_i^*}; \mathcal{M}_i^*, \mathcal{E}_i^*, \sigma_i^*) \) for all \( i \in \{1,...,t+1\} \);
For exculpability attacks, we require that an honest ring member cannot be accused of signing twice with respect to the same tag. So, we also add the corruption ability to the adversary in Definition 5.3, where the adversary can query the system for the partial ring key.

**Definition 5.3.** Exculpability of A Traceable Ring Signature Scheme: Let $\text{TRS}=(\text{System-Setup}, \text{Generate-Key}, \text{Generate-Ring Key}, \text{Sign}, \text{Verify}, \text{Trace-User})$ be a traceable ring signature scheme. Additionally, we set that $k$ is a secure parameter, and $\Pr(B_{E,\text{TRS}}(k,A)=1)$ is the probability that the algorithm $B_{E,\text{TRS}}$ returns 1. Then the advantage that the adversary $A$ breaks $\text{TRS}$ is defined as follows:

$$\text{Adv}^{\text{e,TRS-uf}}_{\text{TRS}}(k,q_r,q_p,q_s,h) = \Pr(B_{E,\text{TRS}}(k,A)=1),$$

where $q_r$ is the maximal number of “Register-User Key” oracle queries, $q_p$ is the maximal number of “Generate-Partial Ring Key” oracle queries, $q_s$ is the maximal number of “Sign” oracle queries and $h$ is the running time of $B$. If the advantage that the adversary breaks $\text{TRS}$ is negligible, then the scheme $\text{TRS}$ is secure.

According to the Definition 5.3, the algorithm $B_{E,\text{TRS}}$ is described as follows:

1. **Setup:** Running $\text{System-Setup}$, $(\text{TRK}, pk_R, spk) \leftarrow \text{System-Setup}(1^k)$, and then $\text{TRK}$ and $pk_R$ are passed to $A$.

2. **Queries:** $A$ makes queries to the following oracles for polynomially many times:
   - **Register-User Key():** Given the public parameters $\text{TRK}$, the oracle outputs the private/public key pair $(sk,pk)$ of a user to $A$.
   - **Generate-Partial Ring Key():** Given the public parameters $\text{TRK}$ and the ring public key $pk_R$, the oracle outputs the partial ring key $psk$.
   - **Sign():** Given the public parameters $\text{TRK}$, the ring public key $pk_R$, the public key list $RLPK$, the message $M$ and the event identifier $\epsilon$, the oracle returns a signature $\sigma$ to $A$, where $\sigma \in \{0,1\}^* \cup \{\bot\}$.

3. **Forgery:** $A$ outputs its forgery, $(M^*, \epsilon^*, \sigma^*)$ for $RLPK^*$. It succeeds if
   - (a): $1 \leftarrow \text{Verify}(\text{TRK}, pk_R, RLPK^*, M^*, \epsilon^*, \sigma^*)$;
   - (b): $A$ did not query $\text{Sign}$ on inputs $RLPK^*$, $M^*$ and $\epsilon^*$;
   - (c): “Linked” $\leftarrow \text{Trace-User}(\text{TRK}, RLPK^*, \{M^*, \sigma^*\}, \{M^', \sigma'^*\}, \epsilon^*)$, where $\sigma'$ is any signature outputted from $\text{Sign}$ on inputs $RLPK^*$, $M'$ and $\epsilon^*$.

**Definition 5.4.** Anonymity of A Traceable Ring Signature Scheme: Let $\text{TRS}=(\text{System-Setup}, \text{Generate-Key}, \text{Generate-Ring Key}, \text{Sign}, \text{Verify}, \text{Trace-User})$ be a traceable ring signature scheme. Additionally, we set that $k$ is a secure parameter, and $\Pr(B_{A,\text{TRS}}(k,A)=1)$ is the probability that the algorithm $B_{A,\text{TRS}}$ returns 1. Then the advantage that the adversary $A$ breaks $\text{TRS}$ is defined as follows:

$$\text{Adv}^{\sigma,\text{TRS-uf}}_{\text{TRS}}(k,q_r,q_p,q_s,h) = \Pr(B_{A,\text{TRS}}(k,A)=1),$$

where $q_r$ is the maximal number of “Register-User Key” oracle queries, $q_p$ is the maximal number of “Generate-Partial Ring Key” oracle queries, $q_s$ is the maximal number of “Sign” oracle queries, $q_s$ is the maximal number of “Generate-Partial Ring Key” oracle queries, $q_s$ is the maximal number of “Sign” oracle queries, and $h$ is the running time of $B$. If the advantage that the adversary breaks $\text{TRS}$ is negligible, then the scheme $\text{TRS}$ is secure.
number of “Sign” oracle queries and $h$ is the running time of $B$. If the advantage that the adversary breaks $TRS$ is negligible, then the scheme $TRS$ is secure.

According to the Definition 5.4, the algorithm $B_{A_{TRS}}$ is described as follows:

1) Setup: Running $\text{System-Setup}$, $(TRK, pk_R, spk) \leftarrow \text{System-Setup}(1^k)$, and then $TRK$ and $pk_R$ are passed to $A$.

2) Queries Phase 1: $A$ makes queries to the following oracles for polynomially many times:

- $\text{Register-User Key}()$: Given the public parameters $TRK$, the oracle outputs the private/public key pair $(sk, pk)$ of a user to $A$.
- $\text{Generate-Partial Ring Key}()$: Given the public parameters $TRK$ and the ring public key $pk_R$, the oracle outputs the partial ring key $psk$.
- $\text{Sign}()$: Given the public parameters $TRK$, the ring public key $pk_R$, the public key list $RL.PK$, the message $\mathfrak{m}$ and the event identifier $\mathcal{E}$, the oracle returns a signature $\sigma$ to $A$, where $\sigma \in \{0, 1\}^* \cup \{\bot\}$.

3) Challenge: $A$ sends to the challenger its forgeries, the public keys $pk_0^* \text{ and } pk_1^*$ of two ring members and the tuple $(\mathfrak{m}^*, \mathcal{E}^*, RL.PK^* \cup \{pk_0^*\} \cup \{pk_1^*\})$. The forgeries satisfy the condition that $A$ did not query $\text{Sign}$ on input $pk_0^*$ (and $pk_1^*$).

The challenger picks a random bit $x \in \{0, 1\}$, and then runs and outputs $\sigma^* \leftarrow \text{Sign}(TRK, pk_R, csk_{pk_R}^*, RL.PK^* \cup \{pk_0^*\} \cup \{pk_1^*\}, \mathfrak{m}^*, \mathcal{E}^*)$ to $A$.

4) Queries Phase 2: $A$ makes queries to the following oracles for polynomially many times:

- $\text{Register-User Key}()$: Given the public parameters $TRK$, the oracle outputs the private/public key pair $(sk, pk)$ of a user to $A$.
- $\text{Generate-Partial Ring Key}()$: Given the public parameters $TRK$ and the ring public key $pk_R$, the oracle outputs the partial ring key $psk$.
- $\text{Sign}()$: Given the public parameters $TRK$, the ring public key $pk_R$, the public key list $RL.PK$, the message $\mathfrak{m}$ and the event identifier $\mathcal{E}$, the oracle returns a signature $\sigma$ to $A$, where $\sigma \in \{0, 1\}^* \cup \{\bot\}$, and $A$ did not query $\text{Sign}$ on inputs $pk_0^*$ and $\mathcal{E}^*$ (and $pk_1^*$ and $\mathcal{E}^*$).

5) Guess: $A$ outputs a bit $x' \in \{0, 1\}$ and succeeds if $x' = x$.

6. Traceable ring signature scheme

In the section, we show a traceable ring signature scheme under our framework for $TRS$. Let $TRS=\langle \text{System-Setup}, \text{Generate-Key}, \text{Generate-Ring Key}, \text{Sign}, \text{Verify}, \text{Trace-User} \rangle$ be a traceable ring signature scheme. In $TRS$, all algorithms are described as follows:

1) $TRS$.\text{System-Setup}: The algorithm run by the trusted authority inputs a security parameter $1^k$. Then, let $G_1$ be group of prime order $q$ and module $p$, and $g$ be a generator of $G_1$. The size of the group is determined by the security parameter. And three hash functions, $H_1 : G_1 \rightarrow G_1$ and $H_2 : G_1^* \rightarrow \mathbb{Z}_q$ and $H_3 : G_1^* \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ can be defined. The algorithm outputs the public parameters $TRK = (G_1, q, H_1, H_2, H_3)$.

Then the algorithm randomly chooses $spk \in \mathbb{Z}_q^*$, outputs a ring master key $spk$ to the trusted authority, and then computes and publishes the ring public key $pk_R = g^{spk}$.

2) $TRS$.\text{Generate-Key}: The algorithm run by a ring member generates his public/private key pair $(pk_l, sk_l)$ with $l \in \{1, 2, \ldots, n\}$, where $n$ is the maximal
number of users in a ring. The algorithm randomly chooses $sk_l \in \mathbb{Z}_q^*$, and then computes $pk_l = g^{sk_l}$.

3): TRS.\textit{Generate-Ring Key}: The algorithm run by a ring member inputs $TRK$, and then the following steps are finished:

a): The algorithm run by the trusted authority randomly chooses $a \in \mathbb{Z}_q^*$, computes

$$
\begin{align*}
  u_1 &= g^a, \quad h_3 = H_1(u_1), \\
  x_1 &= h_1^{spk}, \quad v_1 = h_1^a, \\
  c_1 &= H_2(u_1, x_1, v_1, pk_R), \quad r = a + c_1 \cdot spk.
\end{align*}
$$

The algorithm outputs a partial ring key $psk = (x_1, c_1, r)$ to the ring member\(^8\) whose public key is $pk_l$, where $u_1$ is used to trace the real signer.

b): The algorithm run by the ring member with the public key $pk_l$ and the private key $sk_l$ verifies the partial ring key $psk = (x_1, c_1, r)$ by the following computations:

$$
\begin{align*}
  u'_1 &= g^r \cdot (pk_R)^{-c_1}, \\
  h'_1 &= H_1(u'_1), \quad v'_1 = (h'_1)^r \cdot (x_1)^{-c_1}, \\
  c'_1 &= H_2(u'_1, x_1, v'_1, pk_R),
\end{align*}
$$

and then checks $c'_1 = c_1$. If the equation $c'_1 = c_1$ is correct, the ring member accepts $psk$, otherwise the ring member requires that the trusted authority must resend $psk$. Finally, the algorithm computes $ck_l = r + c_1 \cdot sk_l = a + c_1 \cdot (spk + sk_l)$ and outputs the ring key $csk_l = (ck_l, c_1)$ to the ring member for signing\(^9\), and saves $u'_1$ and $psk = (x_1, c_1, r)$, where $u'_1 = u_1 = g^a$.

4): TRS.\textit{Sign}: A ring member with the ring key $csk_l$ needs to sign a message $M \in \{0, 1\}^*$ on an event identifier $E \in \{0, 1\}^*$. The algorithm run by the ring member inputs ($TRK$, $csk_l$, $RL.PK$, $M$, $E$), and then randomly chooses $k, f \in \mathbb{Z}_q^*$, computes\(^10\)

$$
\begin{align*}
  u_2 &= g^k \cdot \prod_{i=1}^{n} pk_i, \quad h_2 = H_1(u_2), \quad x_2 = h_2^{ck_l \cdot f}, \\
  v_2 &= h_2^k, \quad c''_l = c_1 \cdot f, \quad c_2 = H_3(u_2, x_2, v_2, pk_R, c''_l, M, E), \\
  y &= k + c_2 \cdot f \cdot ck_l, \quad u'_1 = (u'_1 \cdot pk_R^{c_2})^{-c_2 \cdot f}.
\end{align*}
$$

Finally, the algorithm outputs a signature $\sigma = \{u''_1, c''_l, c_2, x_2, y\}$, where $pk_l \in RL.PK$.

5): TRS.\textit{Verify}: The signature receivers verify a ring signature $\sigma$. The algorithm run by a signature verifier inputs ($TRK$, $RL.PK$, $M$, $E$, $\sigma$), and computes the following equations:

$$
\begin{align*}
  u'_2 &= g^y \cdot u''_1 \cdot (pk_R)^{-c''_l \cdot c_2} \cdot \prod_{i=1}^{n} pk_i, \\
  h'_2 &= H_1(u'_2), \quad v'_2 = (h'_2)^y \cdot (x_2)^{-c_2}, \\
  c'_2 &= H_3(u'_2, x_2, v'_2, pk_R, c''_l, M, E),
\end{align*}
$$

and then checks $c'_2 = c_2$. If the equation $c'_2 = c_2$ is correct, then the algorithm outputs the boolean value \textbf{accept}, otherwise the algorithm outputs the boolean value \textbf{reject}.

6): TRS.\textit{Trace-User}: The trusted authority traces a ring member (signer) by two valid traceable ring signatures $\sigma_1$ on $M_1$ and $\sigma_2$ on $M_2$ when the

---

\(^8\)When a ring member registers to the ring signature system, the trusted authority must verify his identity by his public key with respect to his private key.

\(^9\)Our proposed scheme partially employs the construction of certificateless signature to solve the problem that the trusted authority is partially trusted.

\(^10\)may be also seen as $\{0, 1\}^*$ in the computation of $H_3()$.
signatures need to be arbitraged. The algorithm run by the trusted authority inputs \( (THK, RL_PK, \{ M_1, \Phi_1 \}, \{ M_2, \Phi_2 \}, E) \), and then the following steps are finished:

**a):** For any potential public key \( pk_{i,1} \in RL_PK \) and the tuple \( \{ M_1, \sigma_1 \} \), the algorithm computes the equation:

\[
(\frac{u_{i,1}''}{(u_{i,1})^2}) \cdot \frac{1}{c_2} = \left(\left((u_{i,1}'(pk_{i,1})^{c_1})^{-c_2} \cdot f \right) \cdot \frac{1}{c_2}ight) = \frac{(u_{i,1}'(pk_{i,1})^{c_1})}{(u_{i,1})^2} = pk_{i,1}.
\]

If the above equation is correct, then the algorithm securely records the public key \( pk_{i,1} \) of the real signer, otherwise if the algorithm does not find the corresponding public key, the algorithm aborts; similarly, the same computation is finished for any potential identity \( pk_{i,2} \in RL_PK \) and the tuple \( \{ M_2, \Phi_2 \} \), and then the algorithm securely records the public key \( pk_{i,2} \) of the real signer, otherwise the algorithm aborts.

**b):** The algorithm outputs the following results according to the comparisons:

- \( \text{Result} = \text{"Independent"}, \) if \( pk_{i,1} \neq pk_{i,2} \);
- \( \text{Result} = \text{"Linked"}, \) else if \( M_1 = M_2 \);
- \( \text{Result} = \text{"} pk_{i,1} \text{"}, \) otherwise.

7. Analysis of the proposed scheme

7.1. Efficiency. In the proposed scheme, \( \sigma = \{ u_{i,1}', c_1', c_2, x_2, y \} \), where \( u_{i,1}' = (u_{i,1}' \cdot pk_{i,1})^{-c_2} \cdot f \), \( c_1' = c_1 \cdot f \), \( c_2 = H_3(u_{i,1}', x_2, k, pk_{i,1}, c_1, \Phi_1, M, E) \), \( x_2 \) and \( y \) is the number of users in \( G_1 \) and \( |Z_q^*| \) is the size of element in \( G_1 \) and \( |Z_q^*| \). Additionally, the signing and verifying procedure is mainly based on integer multiplication and hash computation, so if we assume that the time for integer multiplication and hash computation can be ignored, then signing a message for a ring signature only needs to compute 5 exponentiations in \( G_1 \) and \( n + 1 \) multiplications in \( G_1 \), and verification requires at most 4 exponentiations in \( G_1 \) and \( n + 3 \) multiplications in \( G_1 \), where \( n \) is the number of users in the ring.

In this paper, we compare the proposed scheme (the scheme of Section 6) with other traceable (or linkable) ring signature schemes proposed by \([4, 25, 26, 40, 55]\). Table 1 shows the performance comparisons of the traceable or linkable ring signature schemes\(^{11}\). Compared with other ring signature schemes, our scheme does not employ pairing computation, and has some advantages in signing and verification costs. Further, the signature size and the linking cost both are constant in our scheme. Table 2 shows some other comparisons of the schemes.

**Table 1. Performance comparisons of the Six Schemes**

| Scheme | Signature Size | Signing Cost | Verification Cost |
|--------|----------------|--------------|-------------------|
| Scheme [40] | \( O(n) \) | \( (4 \cdot n + 3) \cdot e_1 + 2 \cdot n \cdot m_1 \) | \( 4 \cdot n \cdot e_1 + n \cdot m_1 \) |
| Scheme [55] | \( O(n) \) | \( (8 \cdot n + 9) \cdot m_3 + 2 \cdot n + 14 \cdot a \) | \( 28 \cdot n \cdot m_3 + 19 \cdot n \cdot a \) |
| Scheme [25] | \( O(c) \) | \( n + 9 \cdot e_1 + (n + 2) \cdot m_1 \) | \( 2 \cdot n + 3 \cdot e_1 + 2 \cdot n \cdot m_1 + 9 \cdot p \) |
| Scheme [26] | \( O(n) \) | \( (5 \cdot n + 1)^2 + (3 \cdot n + 2) \cdot m_1 \) | \( 5 \cdot n \cdot e_1 + 3 \cdot n \cdot m_1 \) |
| Scheme [4] | \( O(1) \) | \( 7 \cdot e_1 + 7 \cdot m_1 \) | \( 9 \cdot e_1 + 5 \cdot m_1 + 7 \cdot e_2 + 8 \cdot m_2 + 12 \cdot p \) |
| Our Scheme | \( O(1) \) | \( 5 \cdot e_1 + (n + 1) \cdot m_1 \) | \( 4 \cdot e_1 + (n + 3) \cdot m_1 \) |

\(^{11}\)In Table 1, \( e_1 \) is an exponentiation in \( G_1 \), \( m_1 \) is a multiplication in \( G_1 \), \( e_2 \) is an exponentiation in \( G_2 \), \( m_2 \) is a multiplication in \( G_2 \), \( a \) is an addition in \( Z_q^* \), \( m_3 \) is a multiplication in \( Z_q^* \) and \( p \) is a pairing computation.
Table 2. Other comparisons of the Six Schemes

| Scheme | Cryptography | Traceability | Model |
|--------|--------------|--------------|-------|
| [40]   | Public Key   | No           | random oracle |
| [55]   | Public Key   | No           | random oracle |
| [24]   | Public Key   | Yes          | without random oracle |
| [26]   | Public Key   | Yes          | random oracle |
| [4]    | Identity-based | Yes        | random oracle |
| Our Scheme | Public Key   | Yes          | random oracle |

7.2. Security. In the section, we show the proposed scheme (the scheme of Section 6) has the security reduction to the CDH and DDH assumptions and the TRS unforgeability (against linkability attacks and exculpability attacks) under the adaptive chosen message attacks, and has the TRS anonymity. Our proofs for the following theorems are based on the security model of Section 5 (We defer the proofs to Appendix A). In addition, although the unforgeability may be proved by jointing the Theorem 7.2 and the Theorem 7.3 (see [4,26] for more details), we still show the proof of Theorem 7.1.

**Theorem 7.1.** The scheme of Section 6 is \((h, \varepsilon, q_r, q_p, q_s)-unforgeable\) (according to the Definition 5.1), assuming that the \((h', \varepsilon')-CDH\) assumption holds in \(G_1\), where

\[
\varepsilon' = \varepsilon - \frac{2^{-\frac{1}{2}q_n}}{-2^{-\frac{1}{2}q_n}} - \frac{2^{-\frac{1}{2}q_n} + q_s}{2^{-\frac{1}{2}q_n}} - \frac{1}{2^{-\frac{1}{2}q_n} - \frac{q_n}{2\cdot 2^{-\frac{1}{2}q_n}}},
\]

\[
h' = h + O((q_h + q_r + 3 \cdot q_p + 11 \cdot q_s) \cdot C_{exp} + q_s \cdot (6 + n) \cdot C_{mul}),
\]

and \(q_n\) is the maximal number of “Hash” oracle queries, \(q_s\) is the maximal number of “Register-User Key” oracle queries, \(q_p\) is the maximal number of “Generate-Partial Ring Key” oracle queries, \(q_s\) is the maximal number of “Sign” oracle queries, \(C_{mul}\) and \(C_{exp}\) are respectively the time for a multiplication and an exponentiation in \(G_1\), \(n\) is the maximal number of users in a ring.

**Theorem 7.2.** The scheme of Section 6 is a linkable (traceable) TRS scheme when it satisfies the following condition (according to the Definition 5.2)—the scheme of Section 6 is \((h, \varepsilon, q_r, q_p, q_s)-secure\), assuming that the \((h', \varepsilon')-CDH\) assumption holds in \(G_1\), where

\[
\varepsilon' = \varepsilon - \frac{2^{-\frac{1}{2}q_n}}{-2^{-\frac{1}{2}q_n}} - \frac{2^{-\frac{1}{2}q_n} + q_s}{2^{-\frac{1}{2}q_n}} - \frac{1}{2^{-\frac{1}{2}q_n} - \frac{q_n}{2\cdot 2^{-\frac{1}{2}q_n}}},
\]

\[
h' = h + O((q_h + q_r + 3 \cdot q_p + 11 \cdot q_s) \cdot C_{exp} + q_s \cdot (6 + n) \cdot C_{mul}),
\]

and \(q_n\) is the maximal number of “Hash” oracle queries, \(q_s\) is the maximal number of “Register-User Key” oracle queries, \(q_p\) is the maximal number of “Generate-Partial Ring Key” oracle queries, \(q_s\) is the maximal number of “Sign” oracle queries, \(C_{mul}\) and \(C_{exp}\) are respectively the time for a multiplication and an exponentiation in \(G_1\), \(t\) is the number of user (ring member) private key possessed by adversary.

**Theorem 7.3.** The scheme of Section 6 is exculpable when it satisfies the following condition (according to the Definition 5.3)—the scheme of Section 6 is \((h, \varepsilon, q_r, q_p, q_s)-secure\), assuming that the \((h', \varepsilon')-CDH\) assumption holds in \(G_1\), where

\[
\varepsilon' = \varepsilon - \frac{2^{-\frac{1}{2}q_n}}{-2^{-\frac{1}{2}q_n}} - \frac{2^{-\frac{1}{2}q_n} + q_s}{2^{-\frac{1}{2}q_n}} - \frac{1}{2^{-\frac{1}{2}q_n} - \frac{q_n}{2\cdot 2^{-\frac{1}{2}q_n}}},
\]

\[
h' = h + O((q_h + q_r + 3 \cdot q_p + 11 \cdot q_s) \cdot C_{exp} + q_s \cdot (6 + n) \cdot C_{mul}),
\]

12In this paper, we only set \(t = 1\) to make our proof easier to understand. Without loss of generality, our proof may be extended to the generalized situation with \(t > 1\).
and $q_h$ is the maximal number of “Hash” oracle queries, $q_r$ is the maximal number of “Register-User Key” oracle queries, $q_p$ is the maximal number of “Generate-Partial Ring Key” oracle queries, $q_s$ is the maximal number of “Sign” oracle queries, $C_{\text{mul}}$ and $C_{\text{exp}}$ are respectively the time for a multiplication and an exponentiation in $G_1$.

**Theorem 7.4.** The scheme of Section 6 is $(h, \varepsilon, q_r, q_p, q_s)^{13}$-anonymous (according to the Definition 5.4), assuming that the $(h', \varepsilon')$-DDH assumption holds in $G_1$, where

$$
\varepsilon' = \varepsilon - \frac{(q_{r1} + q_{r2}) q_h}{2^n q} - \frac{(q_{s1} + q_{s2})}{2^n q} \cdot \frac{1}{2^n q} - \frac{q_h}{2^n q} - \frac{1}{2},
$$

$h' = h + O((2 q_h + q_{r1} + q_{r2} + 3 q_p) + 11(q_{s1} + q_{s2}) + 12 + (6 + n) C_{\text{mul}})$; $q_h$ is the maximal number of “Hash” oracle queries, $q_{r1}$ and $q_{r2}$ are respectively the maximal numbers of “Register-User Key” oracle queries in the Queries Phase 1 and 2, $q_{p1}$ and $q_{p2}$ are respectively the maximal numbers of “Generate-Partial Ring Key” oracle queries in the Queries Phase 1 and 2, $q_{s1}$ and $q_{s2}$ are respectively the maximal numbers of “Sign” oracle queries in the Queries Phase 1 and 2, $C_{\text{mul}}$ and $C_{\text{exp}}$ are respectively the time for a multiplication and an exponentiation in $G_1$.

8. Conclusions

In this paper, we present a fully traceable ring signature scheme without pairings, which has the security reduction to the computational Diffie-Hellman and decisional Diffie-Hellman assumptions. Also, we give a formal security model for traceable ring signature. Under our security model, the proposed scheme is proved to have the properties of anonymity and traceability with enough security. Compared with other traceable ring signature schemes, the proposed scheme does not employ pairing computation and has some computational advantages. However, because the proposed scheme is still constructed in the random oracle model, the work about TRS still needs to be further progressed.

**Appendix**

(Proof of Theorem 7.1).

**Proof.** Let TRS be a traceable ring signature scheme of Section 6. Additionally, let $A$ be an $(h, \varepsilon, q_r, q_p, q_s)$-adversary attacking TRS. From the adversary $A$, we construct an algorithm $B$, for $(g, g^a, g^b) \in G_1$, the algorithm $B$ is able to use $A$ to compute $g^{a-b}$. Thus, we assume the algorithm $B$ can solve the CDH with probability at least $\varepsilon'$ and in time at most $h'$, contradicting the $(h', \varepsilon')$-CDH assumption. Such a simulation may be created in the following way:

**Setup:** The algorithm inputs a security parameter $1^k$. Then, let $G_1$ be group of prime order $q$ and module $p$, and $g$ be a generator of $G_1$. The size of the group is determined by the security parameter. Also, $H_1 : G_1 \to G_1$, $H_2 : G_1^4 \to \mathbb{Z}_q^*$ and $H_3 : G_1^4 \times \{0, 1\}^* \to \mathbb{Z}_q^*$ can be simulated by the algorithms $H_1$ Queries, $H_2$ Queries and $H_3$ Queries, where we set that $g^b$ ($B$ does not know $b$) is used to answer the query on $H_2$. Additionally, we assume that the user $u^*$ is a challenger, whose public key is $pk^* = g^a$ ($B$ does not know $a$ where $a$ is seen as the corresponding private key). The algorithm outputs the public parameters $TRK= (G_1, g)$. Then the algorithm randomly chooses $d \in \mathbb{Z}_q^*$ and outputs a ring master key $d$, and then computes and publishes the ring public key $pk_R = g^d$.

---

$13\ q_r = q_{r1} + q_{r2}; q_p = q_{p1} + q_{p2}; q_s = q_{s1} + q_{s2}$. 

---

Advances in Mathematics of Communications Volume 14, No. 2 (2020), 207–232
Queries: When running the adversary $A$, the relevant queries can occur according to the Definition 5.1. The algorithm $B$ answers these in the following way:

- **H.1 Queries**: If this query is fresh, then the algorithm chooses random $s \in \mathbb{Z}_q^*$, computes and outputs $(g^h)^s = g^{hs}$ to the adversary $A$; otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple $(s, g^{hs})$ to $U$.

- **H.2 Queries**: If this query is fresh, then the algorithm outputs the new result to the adversary $A$; otherwise the algorithm returns the same result.

- **Register-User Key Queries**: Given the public parameters $TRK$, the algorithm $B$ randomly chooses $sk = \{s, g^s\}$, and then computes the public key $pk = g^s$. The algorithm $B$ outputs the private/public key pair $(sk, pk)$ to $A$, and saves the tuple $(sk, pk)$ to $K$.

- **Generate-Partial Ring Key**

- **Sign Queries**: Given the public parameters $TRK$ and the public key $pk_R$, the algorithm $B$ randomly chooses $t \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t, h_1 = H_1(u_1), x_1 = h_1^{d'}, v_1 = h_1^{d'}, c_1 = H_2(u_1, x_1, v_1, pk_R), r = t + c_1 \cdot d.$$

The algorithm outputs a partial ring key $psk = (x_1, c_1, r)$.

- **Sign Queries**: Given the public parameters $TRK$, the ring public key $pk_R$, the public key list $RLPK$ ($pk_l \in RLPK$ where $pk_l$ is the public key of the ring member that belongs to this ring), the message $M$ and the event identifier $E$, the following setups are finished:

  a): The algorithm randomly chooses $t \in \mathbb{Z}_q^*$, computes

  $$u_1 = g^t, h_1 = H_1(u_1), x_1 = h_1^{d'}, v_1 = h_1^{d'}, c_1 = H_2(u_1, x_1, v_1, pk_R).$$

  b): The algorithm randomly chooses $c_2, y, f \in \mathbb{Z}_q^*$, computes

  $$u_2 = g^y \cdot (g^{c_2 \cdot (pk_l)} - c_2 \cdot f) \cdot (pk_l)^{-c_1 \cdot c_2 \cdot f} \cdot \prod_{i=1}^n pk_i,$$

  where $pk_l \in RLPK$, and then queries the oracle **H.1 Queries** for $u_2$, if $u_2$ has been queried, then the algorithm aborts; otherwise the algorithm continues.

  c): The algorithm randomly chooses $j \in \mathbb{Z}_q^*$, computes

  $$x_2 = (g^t \cdot g^{c_1 \cdot d} \cdot (pk_l)^{c_1})^{j \cdot f} = (g^{t \cdot f} \cdot g^{c_1 \cdot d} \cdot g^{sk_l \cdot c_1})^{j \cdot f},$$

  where we set $h_2 = H_2(u_2) = g^j$ (satisfy the condition that $DL_{h_2}((x_2)^{j \cdot f}) = DL(g^{t \cdot f} \cdot g^{c_1 \cdot d} \cdot g^{sk_l \cdot c_1}) = t + c_1 \cdot d + c_1 \cdot sk_l)$.

  d): The algorithm computes $c_2 = h_3^y \cdot f^c_2$, $u_2'' = (u_1 \cdot pk_l^{c_1})^{-c_2 \cdot f}$ and $c_1'' = c_1 \cdot f$. Then, the algorithm queries the oracle **H.3 Queries**, if the tuple $(u_2, x_2, v_2, pk_R, c_1, f, M, E)$ has been queried, then the algorithm aborts; otherwise the algorithm continues.

  e): The algorithm outputs a ring signature $\sigma = \{u_1', c_1', c_2, x_2, f\}$ to the adversary $A$, and saves the tuple $(t, c_1, f)$ to $S$.

Forgery: If the algorithm $B$ does not abort as a consequence of one of the queries above, the adversary $A$ will, with probability at least $\varepsilon$, return a forgery $(M^*, E^*, \sigma^*, RLPK^*)$ for the challenger $u^*$, where $pk^* \in RLPK^*$. And the forgery satisfies the following condition:

(a): $1 \leftarrow \text{Verify}(TRK, pk_R, RLPK^*, M^*, E^*, \sigma^*)$;

(b): $A$ did not query **Sign** on inputs $RLPK^*$, $M^*$ and $E^*$.
Then, if the adversary $\mathcal{A}$ did not query the oracle $\mathbf{H.1 Queries}$, or $U_{List}$ is empty, or $K_{List}$ is empty or $S_{List}$ is empty, then the algorithm $\mathcal{B}$ aborts.

Otherwise, the algorithm $\mathcal{B}$ can get $h_2 = H_1(*g) = g^{b\cdot s}$. So, when the condition $DL_{h_1}((x_2)^T) = DL_g(g^t \cdot g^{c_1\cdot d} \cdot g^{a\cdot c_1}) = t + c_1 \cdot d + c_1 \cdot a$ holds and $x_2$ can be changed as follows:

$$x_2 = (g^t \cdot g^{c_1\cdot d} \cdot (pk^s)c_1)^{b\cdot s\cdot f} = (g^t \cdot g^{c_1\cdot d} \cdot g^{a\cdot c_1})^{b\cdot s\cdot f}$$

$$= h_2^t \cdot h_2^{c_1\cdot d} \cdot f \cdot (g^{a\cdot b}c_1\cdot s\cdot f)$$

$$= h_2^t \cdot h_2^{c_1\cdot d} \cdot f \cdot (g^{a\cdot b}c_1\cdot s\cdot f)$$

then $\mathcal{B}$ computes and outputs $(x_2 \cdot h_2^{-(t \cdot f + c_1 \cdot d \cdot f)})^{\frac{1}{s}} = g^{a\cdot b}$, which is the solution to the given CDH problem.

Now, we analyze the probability of the algorithm $\mathcal{B}$ not aborting. For the simulation to complete without aborting, we require that all $\mathbf{Sign}$ queries do not abort. So, if the algorithm $\mathcal{B}$ does not abort, then the following conditions must hold:

(a): All $\mathbf{Sign}$ queries do not abort, then we may get the followings:

- The algorithm may abort in the setup b), namely $u_2$ has been queried on the oracle $\mathbf{H.1 Queries}$. So, as $t, c_2, y, f \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q$, the collision probability of $H_1$ is $q_h \cdot \frac{1}{2^{q_h}} = \frac{q_h}{2^{q_h}}$, then the failure probability of the queries is at most $\frac{q_h}{2^{q_h}}$;
- The algorithm may abort in the setup d), namely the tuple $(u_2, x_2, v_2, pk_R, c_1 \cdot f, \mathfrak{M}, \mathcal{E})$ has been queried on the oracle $\mathbf{H.3 Queries}$. So, as $j \in \mathbb{Z}_q^*$ is uniformly distributed in $\mathbb{Z}_q$, the collision probability of $H_3$ is $(q_h + q_s) \cdot \frac{1}{2^q} = \frac{q_h + q_s}{2^q}$, then the failure probability of the queries is at most $\frac{q_h + q_s}{2^q}$.

(b): There are additional probability conditions for forging a valid signature. We set that $NH$ denotes the event that the adversary $\mathcal{A}$ can forge a valid signature while not querying on the oracle $\mathbf{H.1 Queries}$, and that $NQ$ denotes the event that the adversary $\mathcal{A}$ can forge a valid signature while $DL_{h_2}((x_2)^T) \neq DL_g(g^t \cdot g^{c_1\cdot d} \cdot g^{a\cdot c_1})$. So, if the event $NH \cup NQ$ can occur, then the given CDH problem cannot be solved. We need to compute the probability of generating $NH \cup NQ$, then we can get

$$\Pr(NH \cup NQ) = \Pr(NH \cap \neg NQ) + \Pr(NQ).$$

First, we compute $\Pr(NH \cap \neg NQ)$: The event $\neg NQ$ denotes $DL_{h_2}((x_2)^T) = DL_g(g^t \cdot g^{c_1\cdot d} \cdot g^{a\cdot c_1})$, and the event $NH$ denotes that the adversary $\mathcal{A}$ did not query on the oracle $\mathbf{H.1 Queries}$, so we can get the equation $x_2^{NQ} \cdot (\mathfrak{M}^{c_1 \cdot (x_2)^T})^{-1} = h_2 = H_1(u_2)$. Then, because all variants are picked from $\mathbb{Z}_q$, the probability $\Pr(NH \cap \neg NQ)$ is at most $\frac{1}{2^q}$.

Second, we compute $\Pr(NQ)$: Set $u_2 = g^k \cdot \prod_{i=1}^n pk_i$, $h_2 = H_1(u_2)$, $v_2 = h_2^k$, and $x_2 = h_2^{ck_k} \cdot f' \neq h_2^k \cdot f$, as $\sigma = \{u''_1, c''_1, c_2, x_2, y\}$ is valid, then we can compute

$$u_2 = g^y \cdot u''_1 \cdot (pk^s)_k^{-c''_1 \cdot c_2} \cdot \prod_{i=1}^n pk_i = g^y \cdot (g^{ck_k} \cdot f)^{-c_2} \cdot l_{i=1}^n pk_i,$n

$$h_2 = H_1(u_2)$$

$$v_2 = (h_2^y \cdot (x_2)^{-c_2}, k = y - c_2 \cdot f \cdot ck_k, k' = y - c_2 \cdot f' \cdot ck_k'. $$

\textbf{Advances in Mathematics of Communications} \textbf{Volume 14, No. 2 (2020), 207–232}
So, we get $c_2 = H_3(u_2, x_2, v_2, pk_R, c'_1, \mathcal{M}^*, \mathcal{E}^*) = \frac{k-k'}{c_k \cdot f - c_k \cdot f}$. Because the oracle $\text{H.3 Queries}$ $(H_3 : \mathbb{G}_1^t \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^*)$ is uniformly distributed in $\mathbb{Z}_q$, the probability of generating such $c_2$ from $\text{H.3 Queries}$ is at most $\frac{2}{nq}$.

Therefore, from the above analysis, we get that the algorithm $\mathcal{B}$ can compute $g^{a-b}$ from the forgery as shown above, with probability at least $\varepsilon' = \varepsilon - \frac{q \cdot \log q}{2} - \frac{(q_h + q_p^3 \cdot q_p + 11 \cdot q_p \cdot C_{exp} + q_p \cdot (6 + n) \cdot C_{mul})}{2q}$. The time complexity of the algorithm $\mathcal{B}$ is $h' = h + (q_h + q_p + 3 \cdot q_p + 11 \cdot q_p \cdot C_{exp} + q_p \cdot (6 + n) \cdot C_{mul})$, where $n$ is the maximal number of users in a ring and we assume that the time for integer addition, integer multiplication and hash computation can both be ignored.

Thus, Theorem 7.1 follows. \hfill \Box

(Proof of Theorem 7.2).

Proof. Let $\text{TRS}$ be a traceable ring signature scheme of Section 6. Additionally, let $\mathcal{A}$ be an $(h, \varepsilon, q_p, q_p)$-adversary attacking $\text{TRS}$. From the adversary $\mathcal{A}$, we construct an algorithm $\mathcal{B}$, for $(g, g^a, g^b) \in \mathbb{G}_1$, the algorithm $\mathcal{B}$ is able to use $\mathcal{A}$ to compute $g^{a-b}$. Thus, we assume the algorithm $\mathcal{B}$ can solve the CDH with probability at least $\varepsilon'$ and in time at most $h'$, contradicting the $(h', \varepsilon')$-CDH assumption. Such a simulation may be created in the following way:

**Setup:** The algorithm inputs a security parameter $1^k$. Then, let $\mathbb{G}_1$ be group of prime order $q$ and module $p$, and $g$ be a generator of $\mathbb{G}_1$. The size of the group is determined by the security parameter. Also, $H_1 : \mathbb{G}_1 \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_1^t \rightarrow \mathbb{G}_1$ and $H_3 : \mathbb{G}_1 \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ can be simulated by the algorithms $H_1$ Queries, $H_2$ Queries and $H_3$ Queries, where we set that $g^b$ ($\mathcal{B}$ does not know $b$) is used to answer the query on $H_1$ Queries. Additionally, we assume that the user $u^*$ is a challenger, whose public key is $pk^* = g^a$ ($\mathcal{B}$ does not know $a$ where $a$ is seen as the corresponding private key). The algorithm outputs the public parameters $TRK = (\mathbb{G}_1, g)$. Then the algorithm randomly chooses $d \in \mathbb{Z}_q^*$ and outputs a ring master key $d$, and then computes and publishes the ring public key $pk_R = g^d$.

Additionally, to make our description easier to understand, we only set $t = 1$, namely the adversary $\mathcal{A}$ gets the private key of one user.

**Queries:** When running the adversary $\mathcal{A}$, the relevant queries can occur according to the Definition 5.2. The algorithm $\mathcal{B}$ answers these in the following way:

- **H.1 Queries:** If this query is fresh, then the algorithm chooses random $s \in \mathbb{Z}_q^*$, computes and outputs $(g^b)^s = g^{bs}$ to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple $(s, g^{bs})$ to $U.List$.

- **H.2 Queries:** If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.

- **H.3 Queries:** If this query is fresh, then the algorithm outputs the new result to the adversary $\mathcal{A}$; otherwise the algorithm returns the same result.

- **Register-User Key Queries:** Given the public parameters $TRK$, the algorithm $\mathcal{B}$ randomly chooses $sk \in \mathbb{Z}_q^*$, and then computes the public key $pk = g^{sk}$. The algorithm $\mathcal{B}$ outputs the private/public key pair $(sk, pk)$ to $\mathcal{A}$, and saves the tuple $(sk, pk)$ to $K.List$.

- **Generate-Partial Ring Key:** Given the public parameters $TRK$ and the ring public key $pk_R$, the algorithm $\mathcal{B}$ randomly chooses $t \in \mathbb{Z}_q^*$, computes $u_1 = g^t$, $h_1 = H_1(u_1)$, $x_1 = h_1^d$, $\mathcal{B}$ returns the tuple $(u_1, h_1, x_1)$ to $\mathcal{A}$, and saves the tuple $(u_1, h_1, x_1)$ to $K.List$. 

Advances in Mathematics of Communications. Volume 14, No. 2, 2020, 207–232.
The algorithm outputs a partial ring key $psk = (x_1, c_1, r)$.

- **Sign Queries**: Given the public parameters $TRK$, the ring public key $pk_R$, the public key list $RLPK$ ($pk_l \in RLPK$ where $pk_l$ is the public key of the ring member that belongs to this ring), the message $\mathcal{M}$ and the event identifier $\mathcal{E}$, the following setups are finished:
  
  a): The algorithm randomly chooses $t \in \mathbb{Z}_q^*$, computes
  
  $$u_1 = g^t, \ h_1 = H_1(u_1), \ x_1 = h_1^t, \ v_1 = h_1^r, \ c_1 = H_2(u_1, x_1, v_1, pk_R).$$

  b): The algorithm randomly chooses $c_2, y, f \in \mathbb{Z}_q^*$, computes
  
  $$u_2 = g^{y \cdot (g^t \cdot g^{c_1 \cdot d} - c_2 \cdot f) \cdot (pk_l)^{-c_1 \cdot c_2 \cdot f} \cdot \prod_{i=1}^{n} pk_i},$$
  
  where $pk_l \in RLPK$, and then queries the oracle $H_{1 \text{ Queries}}$ for $u_2$, if $u_2$ has been queried, then the algorithm aborts; otherwise the algorithm continues.

  c): The algorithm randomly chooses $j \in \mathbb{Z}_q^*$, computes
  
  $$x_2 = (g^t \cdot g^{c_1 \cdot d} \cdot (pk_l)^{c_2})^{j \cdot f} = (g^t \cdot g^{c_1 \cdot d} \cdot g^{sk_l \cdot c_2})^{j \cdot f},$$

  where we set $h_2 = H_1(u_2) = g^t$ (satisfy the condition that $DL_{h_2}((x_2)^{j}) = DL_g(g^{t} \cdot g^{c_1 \cdot d} \cdot g^{sk_l \cdot c_2}) = t + c_1 \cdot d + c_1 \cdot s(k_i)$).

  d): The algorithm computes $v_2 = h_2^y \cdot x_2^{-c_2}, \ u''_2 = (u_1 \cdot pk_l^{c_2})^{c_2 \cdot f}$ and $c'_1 = c_1 \cdot f$. Then, the algorithm queries the oracle $H_{3 \text{ Queries}}$, if the tuple $(u_2, x_2, v_2, pk_R, c_1, f, \mathcal{M}, \mathcal{E})$ has been queried, then the algorithm aborts; otherwise the algorithm continues.

  e): The algorithm outputs a ring signature $\sigma = \{u''_2, c'_1, c_2, x_2, y\}$ to the adversary $A$, and saves the tuple $(t, c_1, f)$ to $S.List$.

**Forgery**: If the algorithm $B$ does not abort as a consequence of one of the queries above, the adversary $A$ will, with probability at least $\varepsilon$, return a forgery $(\mathcal{M}^*, \mathcal{E}^*, \sigma^*, RLPK^*)$ for the challenger $u^*$, where $pk^* \in RLPK^*$. And the forgery satisfies the following condition:

(a): $1 \leftarrow \text{Verify}(TRK, pk_R, RLPK^*, \mathcal{M}^*, \mathcal{E}^*, \sigma^*)$;

(b): $A$ did not query $\text{Sign}$ on inputs $RLPK^*, \mathcal{M}^*$ and $\mathcal{E}^*$.

Then, if the adversary $A$ did not query the oracle $H_{1 \text{ Queries}}$, or $U.List$ is empty, or $K.List$ is empty or $S.List$ is empty, then the algorithm $B$ aborts.

Otherwise, the algorithm $B$ can get $h_2 = H_1(\ast) = g^{b \cdot s}$. So, when the condition $DL_{h_2}((x_2)^{j}) = DL_g(g^t \cdot g^{c_1 \cdot d} \cdot g^{a \cdot c_1}) = t + c_1 \cdot d + c_1 \cdot a$ holds and $x_2$ can be changed as follows:

\[
\begin{align*}
  x_2 &= (g^t \cdot g^{c_1 \cdot d} \cdot (pk_l)^{c_2})^{b \cdot s \cdot f} = (g^t \cdot g^{c_1 \cdot d} \cdot g^{a \cdot c_1})^{b \cdot s \cdot f} \\
  &= g^{t \cdot b \cdot s \cdot f} \cdot g^{c_1 \cdot d \cdot b \cdot s \cdot f} \cdot g^{a \cdot c_1 \cdot b \cdot s \cdot f} \\
  &= h_2^{t \cdot f} \cdot h_2^{c_1 \cdot d \cdot f} \cdot h_2^{a \cdot b \cdot c_1 \cdot s \cdot f} \\
  &= h_2^{t \cdot f} \cdot h_2^{c_1 \cdot d \cdot f} \cdot (g^{a \cdot b})^{c_1 \cdot s \cdot f} \\
  &= h_2^{t \cdot f + c_1 \cdot d \cdot f} \cdot (g^{a \cdot b})^{c_1 \cdot s \cdot f},
\end{align*}
\]

then $B$ computes and outputs $(x_2 \cdot h_2^{-(t \cdot f + c_1 \cdot d \cdot f) \cdot c_1 \cdot s \cdot f}) = g^{a \cdot b}$, which is the solution to the given CDH problem.

Similarly, we can analyze the probability of the algorithm $B$ not aborting. Therefore, we get that the algorithm $B$ can compute $g^{a \cdot b}$ from the forgery as shown above, with probability at least $\varepsilon' = \varepsilon - \frac{2q}{2q + q} - \frac{2q(q + q_1)}{2q} - \frac{1}{2q} - \frac{q_1}{2q}$. The time complexity of the algorithm $B$ is $h' = h + O((q_b + q_r + 3 \cdot q_p + 11 \cdot q_k) \cdot C_{exp} + q_s \cdot (6 + n) \cdot C_{mul})$, where $n$ is the maximal number of users in a ring.
Thus, Theorem 7.2 follows.

(Proof of Theorem 7.3).

Proof. Let TRS be a traceable ring signature scheme of Section 6. Additionally, let \( A \) be an \((h, \varepsilon, q_r, q_p, q_q)\)-adversary attacking TRS. From the adversary \( A \), we construct an algorithm \( B \), for \((g, g^a, g^b) \in G_1\), the algorithm \( B \) is able to use \( A \) to compute \( g^{a-b} \). Thus, we assume the algorithm \( B \) can solve the CDH with probability at least \( \varepsilon' \) and in time at most \( h' \), contradicting the \((h', \varepsilon')\)-CDH assumption. Such a simulation may be created in the following way:

**Setup:** The algorithm inputs a security parameter \( 1^k \). Then, let \( G_1 \) be group of prime order \( q \) and module \( p \), and \( g \) be a generator of \( G_1 \). The size of the group is determined by the security parameter. Also, \( H_1 : G_1 \rightarrow G_1 \), \( H_2 : G_1^* \rightarrow \mathbb{Z}_q^* \), and \( H_3 : \mathbb{G}_1^* \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^* \) can be simulated by the algorithms \( H_1 \) Queries, \( H_2 \) Queries and \( H_3 \) Queries, where we set that \( g^b \) (\( B \) does not know \( b \)) is used to answer the query on \( H_1 \) Queries. Additionally, we assume that the user \( u^* \) is a challenger, whose public key is \( pk^* = g^a \) (\( B \) does not know \( a \) where \( a \) is seen as the corresponding private key). The algorithm outputs the public parameters \( TRK = (G_1, g) \). Then the algorithm randomly chooses \( d \in \mathbb{Z}_q^* \) and outputs a ring master key \( d \), and then computes and publishes the ring public key \( pk_R = g^d \).

**Queries:** When running the adversary \( A \), the relevant queries can occur according to the Definition 5.3. The algorithm \( B \) answers these in the following way:

- **H_1 Queries:** If this query is fresh, then the algorithm chooses random \( s \in \mathbb{Z}_q^* \), computes and outputs \((g^b)^s = g^{bs} \) to the adversary \( A \); otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple \((s, g^{bs})\) to \( U, List \).

- **H_2 Queries:** If this query is fresh, then the algorithm outputs the new result to the adversary \( A \); otherwise the algorithm returns the same result.

- **H_3 Queries:** If this query is fresh, then the algorithm outputs the new result to the adversary \( A \); otherwise the algorithm returns the same result.

- **Register-User Key Queries:** Given the public parameters \( TRK \), the algorithm \( B \) randomly chooses \( sk \in \mathbb{Z}_q^* \), and then computes the public key \( pk = g^{sk} \). The algorithm \( B \) outputs the private/public key pair \((sk, pk)\) to \( A \), and saves the tuple \((sk, pk)\) to \( K, List \).

- **Generate-Partial Ring Key:** Given the public parameters \( TRK \) and the ring public key \( pk_R \), the algorithm \( B \) randomly chooses \( t \in \mathbb{Z}_q^* \), computes

\[
\begin{align*}
  u_1 &= g^t, h_1 = H_1(u_1), x_1 = h_1^d, \\
  v_1 &= h_1^t, c_1 = H_2(u_1, x_1, v_1, pk_R), r = t + c_1 \cdot d.
\end{align*}
\]

The algorithm outputs a partial ring key \( psk = (x_1, c_1, r) \).

- **Sign Queries:** Given the public parameters \( TRK \), the ring public key \( pk_R \), the public key list \( RL,PK \) \((pk_i \in RL,PK \) where \( pk_i \) is the public key of the ring member that belongs to this ring\), the message \( M \) and the event identifier \( \mathcal{E} \), the following setups are finished:

  a): The algorithm randomly chooses \( t \in \mathbb{Z}_q^* \), computes

\[
\begin{align*}
  u_1 &= g^t, h_1 = H_1(u_1), x_1 = h_1^d, \\
  v_1 &= h_1^t, c_1 = H_2(u_1, x_1, v_1, pk_R).
\end{align*}
\]

  b): The algorithm randomly chooses \( c_2, y, f \in \mathbb{Z}_q^* \), computes

\[
\begin{align*}
  u_2 &= g^{t \cdot \prod_{i=1}^{n} pk_i}, (g^t \cdot g^{f \cdot d})^{-c_2 \cdot f} \cdot (pk_i)^{-c_1 \cdot c_2 \cdot f} \cdot \prod_{i=1}^{n} pk_i.
\end{align*}
\]
where \( \mathds{r}_k \in RLPK \), and then queries the oracle \texttt{H.1 Queries} for \( u_2 \), if \( u_2 \) has been queried, then the algorithm aborts; otherwise the algorithm continues.

c): The algorithm randomly chooses \( j \in \mathbb{Z}_q^* \), computes
\[
x_2 = (g^{c_1} \cdot g^{c_1 \cdot d} \cdot (\mathds{r}_k)^{c_1})^{j \cdot f},
\]
where we set \( h_2 = H_1(u_2) = g^{j} \) (satisfy the condition that \( DL_{h_2}(x_2) = DL_g(g^{c_1 \cdot d} \cdot g^{sk_i \cdot c_1}) = t + c_1 \cdot d + c_1 \cdot s_{kl} \).

d): The algorithm computes \( v_2 = h_2^{q} \cdot x_2^{c_2} \), \( u'_2 = (u_1 \cdot q^{c_1})^{-c_2 \cdot f} \) and \( c'_1 = c_1 \cdot f \). Then, the algorithm queries the oracle \texttt{H.3 Queries}, if the tuple \((u_2, x_2, v_2, pk_R, c_1, f, \mathcal{M}, \mathcal{E})\) has been queried, then the algorithm aborts; otherwise the algorithm continues.

e): The algorithm outputs a ring signature \( \sigma = \{u'_2, c'_1, c_2, x_2, y\} \) to the adversary \( \mathcal{A} \), and saves the tuple \((t, c_1, f)\) to \( S_{List} \).

**Forgery:** Forgery: If the algorithm \( \mathcal{B} \) does not abort as a consequence of one of the queries above, the adversary \( \mathcal{A} \) will, with probability at least \( \varepsilon \), return a forgery \((\mathcal{M}^*, \mathcal{E}^*, \sigma^*, RLPK^*)\) for the challenger \( u^* \), where \( pk^* \in RLPK^* \). And the forgery satisfies the following condition:

(a): \( 1 \leftarrow \texttt{Verify}(TRK, pk_R, RLPK^*, \mathcal{M}^*, \mathcal{E}^*, \sigma^*) \);

(b): \( \mathcal{A} \) did not query \texttt{Sign} on inputs \( RLPK^*, \mathcal{M}^* \) and \( \mathcal{E}^* \);

(c): “Linked” \( \leftarrow \texttt{Trace-User}(TRK, RLPK^*, \{\mathcal{M}^*, \sigma^*\}, \{\mathcal{M}, \sigma'\}, \mathcal{E}^*) \), where \( \sigma' \) is any signature outputted from \texttt{Sign} on inputs \( RLPK^*, \mathcal{M} \) and \( \mathcal{E}^* \).

Then, if the adversary \( \mathcal{A} \) did not query the oracle \texttt{H.1 Queries}, or \( U_{List} \) is empty, or \( K_{List} \) is empty or \( S_{List} \) is empty, then the algorithm \( \mathcal{B} \) aborts.

Otherwise, the algorithm \( \mathcal{B} \) can get \( h_2 = H_1(*) = g^{j \cdot s} \). So, when the condition \( DL_{h_2}(x_2) = DL_g(g^{c_1 \cdot d} \cdot g^{sk_i \cdot c_1}) = t + c_1 \cdot d + c_1 \cdot a \) holds, \( \mathcal{B} \) can compute and output \( g^{ab} \) as the proof of the Theorem 7.1, which is the solution to the given CDH problem.

Similarly, we can analyze the probability of the algorithm \( \mathcal{B} \) not aborting. Therefore, we get that the algorithm \( \mathcal{B} \) can compute \( g^{ab} \) from the forgery as shown above, with probability at least \( \varepsilon' = \varepsilon - \frac{q + 3}{2 + 2} - \frac{2 \cdot (q_1 + q_2)}{2 + 2} = \frac{-q_1}{2 + 2} \). The time complexity of the algorithm \( \mathcal{B} \) is \( h' = h + O((q + q_2 + 3 \cdot q_2 + 11 \cdot q_3) \cdot C_{exp} + q_s \cdot (6 + n) \cdot C_{mul}) \), where \( n \) is the maximal number of users in a ring.

Thus, Theorem 7.3 follows. \( \square \)

**Proof of Theorem 7.4.**

Proof. Let \( \text{TRS} \) be a traceable ring signature scheme of Section 6. Additionally, let \( \mathcal{A} \) be an \( (h, \varepsilon, q_t, q_s, q_3) \)-adversary attacking \( \text{TRS} \). From the adversary \( \mathcal{A} \), we construct an algorithm \( \mathcal{B} \), for \((g, g^a, g^b, g^{ab})\in \mathbb{G}_1 \), the algorithm \( \mathcal{B} \) is able to use \( \mathcal{A} \) to decide \( a_1 \cdot b \equiv a_0 \cdot b \). Thus, we assume the algorithm \( \mathcal{B} \) can solve the DDH with probability at least \( \varepsilon' \) and in time at most \( h' \), contradicting the \((h', \varepsilon')\)-DDH assumption. Such a simulation may be created in the following way:

**Setup:** The algorithm inputs a security parameter \( 1^k \). Then, let \( \mathbb{G}_1 \) be group of prime order \( q \) and module \( p \), and \( g \) be a generator of \( \mathbb{G}_1 \). The size of the group is determined by the security parameter. Also, \( H_1 : \mathbb{G}_1 \rightarrow \mathbb{G}_2 \), \( H_2 : \mathbb{G}_1^2 \rightarrow \mathbb{G}_2^* \) and \( H_3 : \mathbb{G}_1^1 \times \{0, 1\}^* \rightarrow \mathbb{G}_2^* \) can be simulated by the algorithms \{H1 Queries, H2 Queries and H3 Queries\}, where we set that \( g^b \) (\( \mathcal{B} \) does not know \( b \)) is used to answer the query on \( H_1 \) Queries. The algorithm outputs the public parameters \( \text{TRK} = \{\mathbb{G}_1, g\} \). The algorithm outputs the public parameters \( \text{TRK} = \{\mathbb{G}_1, g\} \). Then the algorithm
randomly chooses $d \in \mathbb{Z}_q^*$ and outputs a ring master key $d$, and then computes and publishes the ring public key $pk_R = g^d$.

Additionally, we assume the users $u_0^*$ and $u_1^*$ are two challengers, whose public keys respectively are $pk_0^*$, and $pk_1^*$. We set that the public key of $u_0^*$, $pk_0^* = g^{a_0}$ ($B$ doesn’t know $a_0$ where $a_0$ is seen as the corresponding private key), and that the public key of $u_1^*$, $pk_1^* = g^{a_1}$ ($B$ doesn’t know $a_1$ where $a_1$ is seen as the corresponding private key).

**Queries Phase 1:** When running the adversary $A$, the relevant queries can occur according to the Definition 5.4. The algorithm $B$ answers these in the following way:

- **H.1 Queries:** If this query is fresh, then the algorithm chooses random $s \in \mathbb{Z}_q^*$, computes and outputs $(g^s)^s = g^{bs}$ to the adversary $A$; otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple $(s, g^{bs})$ to $U.List$.
- **H.2 Queries:** If this query is fresh, then the algorithm outputs the new result to the adversary $A$; otherwise the algorithm returns the same result.
- **H.3 Queries:** If this query is fresh, then the algorithm outputs the new result to the adversary $A$; otherwise the algorithm returns the same result.
- **Register-User Key Queries:** Given the public parameters $TRK$, the algorithm $B$ randomly chooses $sk \in \mathbb{Z}_q^*$, and then computes the public key $pk = g^{sk}$. The algorithm $B$ outputs the private/public key pair $(sk, pk)$ to $A$ and saves the tuple $(sk, pk)$ to $K.List$.
- **Generate-Partial Ring Key:** Given the public parameters $TRK$ and the ring public key $pk_R$, the algorithm $B$ randomly chooses $t \in \mathbb{Z}_q^*$, computes

$$u_1 = g^t, \quad h_1 = H_1(u_1), \quad x_1 = h_1^d, \quad v_1 = h_1^t, \quad c_1 = H_2(u_1, x_1, v_1, pk_R), \quad r = t + c_1 \cdot d.$$ 

The algorithm outputs a partial ring key $psk = (x_1, c_1, r)$.

- **Sign Queries:** Given the public parameters $TRK$, the ring public key $pk_R$, the public key list $RL.PK$ ($pk_i \in RL.PK$ where $pk_i$ is the public key of the ring member that belongs to this ring), the message $\mathcal{M}$ and the event identifier $\mathcal{E}$, the following setups are finished:

  a): The algorithm randomly chooses $t \in \mathbb{Z}_q^*$, computes

  $$u_1 = g^t, \quad h_1 = H_1(u_1), \quad x_1 = h_1^d, \quad v_1 = h_1^t, \quad c_1 = H_2(u_1, x_1, v_1, pk_R).$$

  b): The algorithm randomly chooses $c_2, y, f \in \mathbb{Z}_q^*$, computes

  $$u_2 = g^y \cdot (g^t \cdot g^{c_2 \cdot d})^{-c_2 \cdot f} \cdot (pk_i)^{-c_1 \cdot c_2 \cdot f} \cdot \prod_{i=1}^{n} pk_i,$$

  where $pk_i \in RL.PK$, and then queries the oracle $H.1 Queries$ for $u_2$, if $u_2$ has been queried, then the algorithm aborts; otherwise the algorithm continues.

  c): The algorithm randomly chooses $j \in \mathbb{Z}_q^*$, computes

  $$x_2 = (g^t \cdot g^{c_2 \cdot d} \cdot (pk_i)^{c_1})^{-f} = (g^t \cdot g^{c_2 \cdot d} \cdot g^{sk_i \cdot c_1})^{-f},$$

  where we set $h_2 = H_1(u_2) = g^f$ (satisfy the condition that $DL_{h_2}((x_2)^f) = DL_B(g^t \cdot g^{c_2 \cdot d} \cdot g^{sk_i \cdot c_1} = t + c_1 \cdot d + c_1 \cdot sk_i)$).

  d): The algorithm computes $v_2 = h_2^y \cdot x_2^{-c_2}, \quad u_2'' = (u_1 \cdot pk_i^{c_1})^{-c_2 \cdot f} \cdot v_2$ and $c_1'' = c_1 \cdot f$. Then, the algorithm queries the oracle $H.3 Queries$, if the tuple $(u_2, x_2, v_2, pk_R, c_1, f, \mathcal{M}, \mathcal{E})$ has been queried, then the algorithm aborts; otherwise the algorithm continues.
e): The algorithm outputs a ring signature \( \sigma = \{u''_1, c''_1, c_2, x_2, y\} \) to the adversary \( A \), and saves the tuple \((t, c_1, f)\) to \( S\_List\).

**Challenge:** \( A \) sends to the challengers its forgeries, the public keys \( pk'_0 \) and \( pk'_1 \) and the tuple \((\mathfrak{M}, E^*; RL_PK^* \cup \{pk'_0\} \cup \{pk'_1\})\). The forgeries satisfy the condition that \( A \) did not query \( \text{Sign} \) on input \( pk'_0 \) (and \( pk'_1 \)).

The challengers pick a random bit \( x \in \{0, 1\} \), and then run and output \( \sigma^* \leftarrow \text{Sign}(TRK, pk_R, csk_x, RL_PK^* \cup \{pk'_0\} \cup \{pk'_1\}, \mathfrak{M}, E^*) \) to \( A \).

**Queries Phase 2:** When running the adversary \( A \), the relevant queries can occur according to the Definition 5.4. The algorithm \( B \) answers these in the following way:

- **H_1 Queries:** If this query is fresh, then the algorithm chooses random \( s \in Z_q^* \) computes and outputs \( (g^h)^s = g^{h \cdot s} \) to the adversary \( A \); otherwise the algorithm returns the same result. Also, the algorithm saves the new tuple \((s, g^{h \cdot s})\) to \( U\_List\).

- **H_2 Queries:** If this query is fresh, then the algorithm outputs the new result to the adversary \( A \); otherwise the algorithm returns the same result.

- **Register-User Key Queries:** Given the public parameters \( TRK \), the algorithm \( B \) randomly chooses \( sk \in Z_q^* \), and then computes the public key \( pk = g^{sk} \). The algorithm \( B \) outputs the private/public key pair \((sk, pk)\) to \( A \), and saves the tuple \((sk, pk)\) to \( K\_List\).

- **Generate-Partial Ring Key:** Given the public parameters \( TRK \) and the ring public key \( pk_R \), the algorithm \( B \) randomly chooses \( t \in Z_q^* \), computes

\[
\begin{align*}
u_1 &= g^t, h_1 = H_1(u_1), x_1 = h_1^d, \\
v_1 &= h_1^t, c_1 = H_2(u_1, x_1, v_1, pk_R), r = t + c_1 \cdot d.
\end{align*}
\]

The algorithm outputs a partial ring key \( psk = (x_1, c_1, r) \).

- **Sign Queries:** Given the public parameters \( TRK \), the ring public key \( pk_R \), the public key list \( RL_PK \) \( (pk_l \in RL_PK \text{ where } pk_l \text{ is the public key of the ring member that belongs to this ring}) \), the message \( \mathfrak{M} \) and the event identifier \( E \), the following setups are finished:

  a): The algorithm randomly chooses \( t \in Z_q^* \), computes

\[
\begin{align*}
u_1 &= g^t, h_1 = H_1(u_1), x_1 = h_1^d, \\
v_1 &= h_1^t, c_1 = H_2(u_1, x_1, v_1, pk_R).
\end{align*}
\]

  b): The algorithm randomly chooses \( c_2, y, f \in Z_q^* \), computes

\[
\begin{align*}
u_2 &= g^y \cdot (g^t \cdot g^{c_2 \cdot d})^{-c_2 \cdot f} \cdot (pk_l)^{-c_1 \cdot c_2 \cdot f} \cdot \prod_{i=1}^n pk_i,
\end{align*}
\]

where \( pk_i \in RL_PK \), and then queries the oracle **H_1 Queries** for \( u_2 \), if \( u_2 \) has been queried, then the algorithm aborts; otherwise the algorithm continues.

  c): The algorithm randomly chooses \( j \in Z_q^* \), computes

\[
\begin{align*}x_2 &= (g^t \cdot g^{c_2 \cdot d} \cdot (pk_l)^{c_2})^{c_j} f = (g^t \cdot g^{c_2 \cdot d} \cdot g^{sk_l \cdot c_1})^{c_j} f,
\end{align*}
\]

where we set \( h_2 = H_1(u_2) = g^t \) (satisfy the condition that \( DL_h((x_2)^2) = DL_g(g^t \cdot g^{c_2 \cdot d} \cdot g^{sk_l \cdot c_1}) = t + c_1 \cdot d + c_1 \cdot sk_l) \).

  d): The algorithm computes \( v_2 = h_2^y \cdot x_2^{-c_2}, u''_1 = (u_1 \cdot pk_l^{c_2})^{-c_2 \cdot f} \) and \( c''_1 = c_1 \cdot f \). Then, the algorithm queries the oracle **H_3 Queries**, if the tuple \((u_2, x_2, v_2, pk_R, c_1 \cdot f, \mathfrak{M}, E)\) has been queried, then the algorithm aborts; otherwise the algorithm continues.
e): The algorithm outputs a ring signature $\sigma = \{u''_1, c''_1, c_2, x_2, y\}$ to the adversary $A$, and saves the tuple $(t, c_1, f)$ to $S_list$.

**Guess:** If the algorithm $B$ does not abort as a consequence of one of the queries above, the adversary $A$ will, with probability at least $\varepsilon$, output a bit $x' \in \{0, 1\}$ and succeed ($x' = x$), and must return a valid forgery $(\mathfrak{R}^*, \mathfrak{E}^*, \sigma^*, RL\_PK^*)$ for the challengers $u''_1$ and $u'_1$, where $pk^*_c \in RL\_PK^*$.

Then, if the adversary $A$ did not query the oracle $H_1$ Queries, or $U_list$ is empty, or $K_list$ is empty or $S_list$ is empty, then the algorithm $B$ aborts.

Otherwise, the algorithm $B$ can get $h_2 = H_1(\ast) = g^{b^*}$. So, when the condition $DL_{h_2}((x_2)^{\frac{1}{2}}) = DL_g(g^t \cdot g^{c_1 \cdot d} \cdot g^{a_x \cdot c_1}) = t + c_1 \cdot d + c_1 \cdot a_x^*$, holds and $x_2$ can be changed as follows:

\[
x_2 = (g^t \cdot g^{c_1 \cdot d} \cdot (pk^*_c)^b \cdot f) = (g^t \cdot g^{c_1 \cdot d} \cdot g^{a_x \cdot c_1}) \cdot b \cdot f
\]

\[
= h_2^t \cdot f \cdot h_2^{c_1 \cdot d} \cdot g^{a_x \cdot c_1} \cdot b \cdot f
\]

\[
= h_2^t \cdot f \cdot h_2^{c_1 \cdot d} \cdot (g^{a_x \cdot f}) \cdot b \cdot f
\]

\[
= h_2^t + f \cdot c_1 \cdot d \cdot f \cdot (g^{a_x \cdot f}) \cdot b \cdot f,
\]

then $B$ computes and decides $DL_g((x_2 \cdot h_2^{(t \cdot f + c_1 \cdot d) \cdot f}) \cdot b) = b \cdot a_x^* \cdot b = b \cdot a_x^* \cdot b$, which is the solution to the given DDH problem.

Now, we analyze the probability of the algorithm $B$ not aborting. For the simulation to complete without aborting, we require that all $Sign$ queries do not abort in the Queries Phase 1 and 2. So, if the algorithm $B$ does not abort, then the following conditions must hold:

(a): All $Sign$ queries do not abort in the Queries Phase 1 and 2, then we may get the followings:

- The algorithm may abort in the setup b), namely $u_2$ has been queried on the oracle $H_1$ Queries. So, as $t, c_2, y, f \in \mathbb{Z}_q^*$ are uniformly distributed in $\mathbb{Z}_q^4$, the collision probability of $H_1$ is $q_h \cdot \frac{1}{2q^4} = \frac{q_h}{2q^4}$, then the failure probability of the queries is at most $(\frac{q_h + q_1}{2q})$.
- The algorithm may abort in the setup d), namely the tuple $(u_2, x_2, x_2, y, pkR, c_1 \cdot f, \mathfrak{R}, \mathfrak{E})$ has been queried on the oracle $H_3$ Queries. So, as $j \in \mathbb{Z}_q^4$ is uniformly distributed in $\mathbb{Z}_q^4$, the collision probability of $H_3$ is $(2 \cdot q_h + q_1 + q_2) \cdot \frac{1}{2q} = \frac{2q_h + q_1 + q_2}{2q}$, then the failure probability of the queries is at most $(\frac{2q_h}{2q})$.

(b): There are additional probability conditions for forging a valid signature. We set that $NH$ denotes the event that the adversary $A$ can forge a valid signature while not querying on the oracle $H_1$ Queries, and that $NQ$ denotes the event that the adversary $A$ can forge a valid signature while $DL_{h_2}((x_2)^{\frac{1}{2}}) \neq DL_g(g^t \cdot g^{c_1 \cdot d} \cdot g^{a_x \cdot c_1})$. So, if the event $NH \cup NQ$ can occur, then the given DDH problem cannot be solved. We need to compute the probability of generating $NH \cup NQ$, then we can get

\[
Pr(NH \cup NQ) = Pr(NH \cap \neg NQ) + Pr(NQ).
\]

First, we compute $Pr(NH \cap \neg NQ)$: The event $\neg NQ$ denotes $DL_{h_2}((x_2)^{\frac{1}{2}}) = DL_g(g^t \cdot g^{c_1 \cdot d} \cdot g^{a_x \cdot c_1})$, and the event $NH$ denotes that the adversary $A$ did not query on the oracle $H_1$ Queries, so we can get the equation \( x_2^{\frac{t \cdot f + c_1 \cdot d \cdot f}{2q}} = h_2 = H_1(u_2) \). Then, because all variants are picked from $\mathbb{Z}_q^4$, the probability $Pr(NH \cap \neg NQ)$ is at most $\frac{1}{2q}$.
Second, we compute $\Pr(NQ)$: Set $u_2 = g^k \cdot \prod_{i=1}^{n} pk_i$, $h_2 = H_1(u_2)$, $v_2 = h_2^2$, and $x_2 = h_2^{c_k \cdot f} \neq h_2^2$, as $\sigma = \{u_1', c_1, c_2, x_2, y\}$ is valid, then we can compute

$$u_2 = g^y \cdot u''_1 \cdot (g^d)^{c_1 \cdot c_2} \cdot \prod_{i=1}^{n} pk_i = g^y \cdot (g^{c_k \cdot f})^{c_2} \cdot \prod_{i=1}^{n} pk_i,$$

$$h_2 = H_1(u_2), \quad v_2 = (h_2)^y \cdot (x_2)^{-c_2},$$

$$k = y - c_2 \cdot f \cdot c_k, \quad k' = y - c_2 \cdot f' \cdot c_k.'$$

So, we get $c_2 = H_3(u_2, x_2, v_2, pk_R, c_1', \mathcal{M}^*, \mathcal{E}^*) = \frac{h - k'}{ck_f' - ck_f}$. Because the oracle $H_3$ Queries ($H_3 : \mathbb{G}_1^4 \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$) is uniformly distributed in $\mathbb{Z}_q$, the probability of generating such $c_2$ from $H_3$ Queries is at most $\frac{\theta}{2^{c_k}}$.

Therefore, from the above analysis, we get that the algorithm $B$ can decide $a_1 \cdot b \equiv a_0 \cdot b$ from the forgery as shown above, with probability at least $\varepsilon' = \varepsilon - \frac{(q_1 + q_2) + 1}{2} - \frac{(q_1 + q_2)}{2} - \frac{1}{2} + \frac{1}{2}$. The time complexity of the algorithm $B$ is $h' = h + O((2 \cdot q_1 + q_{r_1} + q_{r_2} + 3 \cdot (q_{p_1} + q_{p_2}) + 11 \cdot (q_{s_1} + q_{s_2})) \cdot C_{\text{exp}} + (q_{s_1} + q_{s_2}) \cdot (6 + n) \cdot C_{\text{mul}})$, where $n$ is the maximal number of users in a ring and we assume that the time for integer addition, integer multiplication and hash computation can both be ignored.

Thus, Theorem 7.4 follows. \qed

**Acknowledgments**

This work is supported by the National Natural Science Foundations of China (No.61402055), the Hunan Provincial Natural Science Foundation of China (No.2018 JJ2445) and the Open Research Fund of Key Laboratory of Network Crime Investigation of Hunan Provincial Colleges (No.2017WLJZZC003).

**References**

[1] M. Abe, M. Ohkubo and K. Suzuki, 1-out-of-n signatures from a variety of keys, Advances in Cryptology—ASIACRYPT 2002, (2002), 415–432.
[2] M. Abe, M. Ohkubo and K. Suzuki, Efficient threshold signer-ambiguous signatures from variety of keys, IEICE Trans 2004, 87 (2004), 471–479.
[3] M. H. Au, S. S. M. Chow, W. Susilo and P. P. Tsang, Short linkable ring signatures revisited, Public Key Infrastructure, (2006), 101–115.
[4] M. H. Au, J. K. Liu, W. Susilo and T. H. Yuen, Secure ID-based linkable and revocable-iff-linked ring signature with constant-size construction, Theoretical Computer Science, 469 (2013), 1–14.
[5] M. H. Au, J. K. Liu, W. Susilo and T. H. Yuen, Constant-size ID-based linkable and revocable-iff-linked ring signature, Progress in cryptology—INDOCRYPT 2006, 4329 (2006), 364–378.
[6] M. H. Au, J. K. Liu, T. H. Yuen and D. S. Wong, ID-based ring signature scheme secure in the standard model, Advances in Information and Computer Security, Lecture Notes in Comput. Sci., Springer, Berlin, 4266 (2006), 1–16.
[7] P. S. L. M. Barreto, B. Libert, N. McCullagh and J.-J. Quisquater, Efficient and provably-secure identity-based signatures and signcryption from bilinear maps, Advances in Cryptology—ASIACRYPT 2005, Lecture Notes in Comput. Sci., Springer, Berlin, 3788 (2005), 515–532.
[8] A. Bender, J. Katz and R. Morselli, Ring signatures: Stronger definitions, and constructions without random oracles, Theory of Cryptography, Lecture Notes in Comput. Sci., Springer, Berlin, 3876 (2006), 60–79.
[9] D. Boneh and M. Franklin, Identity-based encryption from the Weil pairing, Advances in Cryptology—CRYPTO 2001 (Santa Barbara, CA), Lecture Notes in Comput. Sci., Springer, Berlin, 2139 (2001), 213–229.
[10] D. Boneh, C. Gentry, B. Lynn and H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, Advances in Cryptology—EUROCRYPT 2003, Lecture Notes in Comput. Sci., Springer, Berlin, 2656 (2003), 416–432.
[11] D. Boneh and M. Harnag, Generalized identity based and broadcast encryption schemes, Advances in Cryptology—ASIACRYPT 2008, Lecture Notes in Comput. Sci., Springer, Berlin, 5350 (2008), 455–470.
[12] S. Brands, Untraceable off-line cash in wallet with observers, CRYPTO’93, 773 (1993), 302–318.
[13] E. Bresson, J. Stern and M. Szydlo, Threshold ring signatures and applications to ad-hoc groups, Advances in Cryptology—CRYPTO 2002, Lecture Notes in Comput. Sci., Springer, Berlin, 2442 (2002), 465–480.
[14] J. C. Cha and J. H. Cheon, An identity-based signature from gap Diffie-Hellman groups, Public Key Cryptography—PKC 2003, Lecture Notes in Comput. Sci., Springer, Berlin, 2567 (2002), 18–30.
[15] D. Chaum, Blind signatures for untraceable payments, Advances in Cryptology, 397 (1983), 199–203.
[16] D. Chaum, A. Fiat and M. Naor, Untraceable electronic cash, Advances in Cryptology—CRYPTO’88, (1988), 319–327.
[17] D. Chaum and T. P. Pedersen, Wallet databases with observers, In Ernest Brickell, Proceedings of Crypto 92, 0740 (1992), 89–105.
[18] D. Chaum and E. Van Heyst, Group signatures, Advances in Cryptology—EUROCRYPT’91, (1991), 257–265.
[19] B. Chevallier-Mames, An efficient CDH-based signature scheme with a tight security reduction, Advances in Cryptology—CRYPTO 2005, Lecture Notes in Comput. Sci., Springer, Berlin, 3621 (2005), 511–526.
[20] S. S. M. Chow, J. K. Liu and D. S. Wong, Robust receipt-free election system with ballot secrecy and verifiability, NDSS 2008, (1993), 1–14.
[21] S. S. M. Chow, S. M. Yiu and L. C. K. Hui, Efficient identity based ring signature, ACNS 2005, 3531 (2005), 499–512.
[22] I. Damgørd, K. Dupont and M. Pedersen, Uncloneable group identification, Advances in Cryptology—EUROCRYPT 2006, Lecture Notes in Comput. Sci., Springer, Berlin, 4004 (2006), 555–572.
[23] Y. Dodis, A. Kiayias, A. Nicolosi and V. Shoup, Anonymous identification in Ad hoc groups, Advances in Cryptology—EUROCRYPT 2004, Lecture Notes in Comput. Sci., Springer, Berlin, 3027 (2004), 609–626.
[24] K. Emura, A. Miyaji and K. Omote, An r-hiding revocable group signature scheme: Group signatures with the property of hiding the number of revoked users, Journal of Applied Mathematics, 2014 (2011), 1–14.
[25] E. Fujisaki, Sub-linear size traceable ring signatures without random oracles, Topics in Cryptology—CT-RSA 2011, Lecture Notes in Comput. Sci., Springer, Heidelberg, 6558 (2011), 393–415.
[26] E. Fujisaki and K. Suzuki, Traceable ring signature, Public Key Cryptography 2007, Lecture Notes in Comput. Sci., Springer, Berlin, 4450 (2007), 181–200.
[27] E.-J. Goh and S. Jarecki, A signature scheme as secure as the Diffie-Hellman problem, Advances in Cryptology—EUROCRYPT 2003, Lecture Notes in Comput. Sci., Springer, Berlin, 2656 (2003), 401–415.
[28] K. Gu, W. Jia and C. Jiang, Efficient and secure identity-based proxy signature in the standard model, The Computer Journal, 58 (2015), 792–807.
[29] K. Gu, W. J. Jia, G. J. Wang and S. Wen, Efficient and secure attribute-based signature for monotone predicates, Acta Informatica, 54 (2017), 521–541.
[30] K. Gu, W. J. Jia and J. M. Zhang, Identity-based multi-proxy signature scheme in the standard model, Fundamenta Informaticae, 150 (2017), 179–210.
[31] F. Hess, Efficient identity based signature schemes based on pairings, Selected Areas in Cryptography, Lecture Notes in Comput. Sci., Springer, Berlin, 2595 (2003), 310–324.
[32] L. Ibraimi, S. I. Nikova, P. H. Hartel and W. Jonker, An identity-based group signature with membership revocation in the standard model, Faculty of Electrical Engineering, Mathematics & Computer Science, Available from: http://doc.utwente.nl/72270/1/Paper.pdf.
[33] M. Jakobsson and C. P. Schnorr, Efficient oblivious proofs of correct exponentiation, Proceedings of the IFIP Conference on Communications and Multimedia Security 99, 152 (1999), 71–80.
I. R. Jeong, J. O. Kwon and D. H. Lee, Analysis of revocable-iff-linked ring signature scheme, *IEICE Transactions on Fundamentals of Electronics Communications & Computer Sciences*, 92 (2009), 322–325.

Y. Komano, K. Ohta, A. Shimbo and S. Kawamura, Toward the fair anonymous signatures: Deniable ring signatures, *Topics in Cryptology—CT-RSA 2006*, Lecture Notes in Comput. Sci., Springer, Berlin, 3680 (2006), 174–191.

F. Laguillaumie and D. Vergnaud, Multi-designated verifiers signatures, *Information and Communications Security*, (2004), 495–507.

J. K. Liu, M. H. Au, W. Susilo and J. Y. Zhou, Linkable ring signature with unconditional anonymity, *IEEE Transactions on Knowledge and Data Engineering*, 26 (2014), 157–165.

D. Y. W. Liu, J. K. Liu, Y. Mu, W. Susilo and D. S. Wong, Revocable ring signature, *J. Comput. Sci. Tech.*, 22 (2007), 785–794.

J. K. Liu, V. K. Wei and D. S. Wong, Linkable spontaneous anonymous group signature for ad hoc groups, *Information Security and Privacy*, (2004), 325–335.

J. K. Liu and D. S. Wong, Linkable ring signatures: Security models and new schemes, *Computational Science and Its Applications—ICCSA 2005*, (2005), 614–624.

J. K. Liu and D. S. Wong, Enhanced security models and a generic construction approach for linkable ring signature, *Int. J. Found. Comput. Sci.*, 17 (2006), 1403–1422.

M. Naor, Deniable ring authentication, *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Comput. Sci., Springer, Berlin, 2442 (2002), 481–498.

T. Okamoto and K. Ohta, Universal electronic cash, *Advances in Cryptology—CRYPTO’91*, 403 (1991), 324–337.

K. G. Paterson and J. C. N. Schuldt, Efficient identity-based signatures secure in the standard model, *A Information Security and Privacy*, (2006), 207–222.

R. L. Rivest, A. Shamir and Y. Tauman, How to leak a secret, *Advances in Cryptology—ASIACRYPT 2001 (Gold Coast)*, Lecture Notes in Comput. Sci., Springer, Berlin, 2248 (2001), 552–565.

A. Shamir and Y. Tauman, Improved online/offline signature scheme, *Advances in Cryptology—CRYPTO 2001 (Santa Barbara, CA)*, Lecture Notes in Comput. Sci., Springer, Berlin, 2139 (2001), 355–367.

W. Susilo and Y. Mu, Non-interactive deniable ring authentication, *Information Security and Cryptology—ICISC*, Lecture Notes in Comput. Sci., Springer, Berlin, 2971 (2004), 386–401.

P. P. Tsang and V. K. Wei, Short linkable ring signatures for e-voting, e-cash and attestation, *Information Security Practice and Experience*, (2005), 48–60.

P. P. Tsang, V. K. Wei, T. K. Chan, M. H. Au, J. K. Liu and D. S. Wong, Separable linkable threshold ring signatures, *Progress in Cryptology—INDOCRYPT 2004*, Lecture Notes in Comput. Sci., Springer, Berlin, 3348 (2004), 384–398.

B. Waters, Efficient identity-based encryption without random oracles, *Advances in Cryptology—EUROCRYPT 2005*, Lecture Notes in Comput. Sci., Springer, Berlin, 3494 (2005), 114–127.

D. S. Wong, K. Fung, J. K. Liu and V. K. Wei, On the RS-code construction of ring signature schemes and a threshold setting of RST, *Information and Communications Security*, (2003), 34–46.

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

S. K. Zeng, S. Q. Jiang and Z. G. Qin, An efficient conditionally anonymous ring signature in the random oracle model, *Theoretical Computer Science*, 461 (2012), 106–114.

F. G. Zhang and K. Kim, ID-based blind signature and ring signature from pairings, *Advances in Cryptology—ASIACRYPT 2002*, Lecture Notes in Comput. Sci., Springer, Berlin, 2501 (2002), 533–547.

D. Zheng, X. X. Li, K. F. Chen and J. H. Li, Linkable ring signatures from linear feedback shift register, *Emerging Directions in Embedded and Ubiquitous Computing*, (2007), 716–727.