Efficient Verifiable Ring Encryption for Ad Hoc Groups

Joseph K. Liu[1], Patrick P. Tsang[1], and Duncan S. Wong[2]

[1] Department of Information Engineering,
The Chinese University of Hong Kong Shatin, Hong Kong
{ksliu, pktsang3}@ie.cuhk.edu.hk
[2] Department of Computer Science,
City University of Hong Kong, Kowloon, Hong Kong
duncan@cityu.edu.hk

Abstract. We propose an efficient *Verifiable Ring Encryption* (VRE) for ad hoc groups. VRE is a kind of verifiable encryption [16,1,4,2,8] in which it can be publicly verified that there exists at least one user, out of a designated group of n users, who can decrypt the encrypted message, while the semantic security of the message and the anonymity of the actual decryptor can be maintained. This concept was first proposed in [10] in the name of *Custodian-Hiding Verifiable Encryption*. However, their construction requires the inefficient cut-and-choose methodology which is impractical when implemented. We are the first to propose an efficient VRE scheme that does not require the cut-and-choose methodology.

In addition, while [10] requires interaction with the encryptor when a verifier verifies a ciphertext, our scheme is non-interactive in the following sense: (1) an encryptor does not need to communicate with the users in order to generate a ciphertext together with its validity proof; and (2) anyone (who has the public keys of all users) can verify the ciphertext, without the help of the encryptor or any users. This non-interactiveness makes our scheme particularly suitable for ad hoc networks in which nodes come and go frequently as ciphertexts can be still generated and/or verified even if other parties are not online in the course. Our scheme is also proven secure in the random oracle model.

1 Introduction

A *Verifiable Encryption* [16,1,4,2,8] allows a prover to encrypt a message and sends to a receiver such that the ciphertext is publicly verifiable. That is, any verifier can ensure the ciphertext can be decrypted by the receiver yet knowing nothing about the plaintext. There are numerous applications of verifiable encryption. For example, in a publicly verifiable secret sharing scheme [16], a dealer shares a secret with several parties such that a third party can verify that the sharing was done correctly. This can be done by verifiably encrypting each shares under the public key of the corresponding party and proves to the third party that the ciphertext encrypt the correct shares. Another scenario is in a fair exchange environment [1], in which both parties want to exchange some

R. Molva, G. Tsudik, and D. Westhoff (Eds.): ESAS 2005, LNCS 3813, pp. 1–13, 2005.

information such that either each party obtain the other's data, or neither party does. One approach is to let both parties verifiably encrypt their data to each other under the public key of a trusted party and then to reveal their data. If one party refuses to do so, the other can go to the trusted party to obtain the required data. Verifiable encryption can be also applied in revokable anonymous credential [5]. When the administration organization issues a credential, it verifiably encrypts enough information under the public key of the anonymity revocation manager, so that later if the identity of the credential owner needs to be revealed, this information can be decrypted.

In an interactive *Custodian-Hiding Verifiable Encryption* (CHVE) [10], an *Encryptor* wants to send a public-key encrypted message to one among a group of n *users* through a *Verifier*. The Encryptor plays the role of a *Prover* and conducts an interactive protocol with the Verifier such that, if the Verifier is satisfied, at least one of the n possible decryptors can recover the message. At the same time, the message is semantically secure, even against the Verifier, and the identity of the actual decryptor is anonymous, again even to the Verifier. Custodian-Hiding Verifiable Encryption can be found useful in the applications of gateway system or receiver-oblivious transfer.

In ad hoc networks, nodes are highly dynamic and may switch from being online and being offline frequently from time to time. The verifiability of interactive Custodian-Hiding Verifiable Encryption schemes is virtually of no practical use if the encryptor goes offline (or leaves the networks forever) since no one can verify the validity of the ciphertext without the help of the encryptor. In the environment of ad hoc networks in which most users are highly mobile, it is unreasonable to require an encryptor to be always online and available to be contacted by a verifier. What we need is exactly a non-interactive approach to verify the ciphertext.

Let us spare a few words explaining the decision of naming our scheme as "Verifiable Ring Encryption" over "Custodian-Hiding Verifiable Encryption", as suggested by [10]. The word "Ring" is borrowed from Ring signatures [15] which is a signature scheme constructed in the structure of a ring in order to achieve 1-out-of-n anonymity of the signer. Analogously, Verifiable Ring Encryption implies an encryption scheme constructed in the structure of a ring, in which ciphertexts can be verified to be decryptable by some one, with the identity of that genuine decryptor hidden among a group of n members. Our choice of "Verifiable Ring Encryption" therefore better conveys the information on what the scheme actually does. Moreover, the non-interactiveness of our scheme suggests that a verifier is convinced by verifying the validity of some kind of proofs. These proofs can actually be thought of a kind of ring signatures in the sense that they convince verifiers of the fact that some 1 out of n users can decrypt a ciphertext, and yet hiding that decryptor's identity.

Finally we would like to note that the notion of "Verifiable Group Encryption" (VGE) has been used by [4] to mean something related but very different: VGE allows the prover to prove that any subset of t members of a group of n users can jointly recover the message behind a ciphertext, by making use of a secret sharing scheme. That is, the prover divides the message into n pieces of

shares such that any t of them are enough to reconstruct m. Then he encrypts each share for each user using the user's encryption function, and sends all ciphertexts to the verifier. It is clear that the message m can be reconstructed if any t users decrypt their corresponding ciphertext to get the shares.

1.1 Contributions

We propose an efficient Verifiable Ring Encryption for ad hoc networks which is the first of its kind that is without the use of the inefficient cut-and-choose methodology. Furthermore, our proposed scheme is non-interactive. Unlike the previous one proposed in [10], in our scheme an encryptor does not need to communicate with the users in order to generate a ciphertext together with its validity proof. Also anyone who has got the public keys of all users can verify the ciphertext without the help of the encryptor or any users. Note that being non-interactive makes our scheme well-suited for ad hoc networks in which nodes are highly mobile. Ciphertexts can be still generated and/or verified even if other parties are not online in the course. We also prove the security of our proposed scheme in the random oracle model [3].

Organization: The rest of the paper is organized as follows. We give security definitions in Sec. 2. The details of our proposed scheme is presented in Sec. 3. Its security is analyzed in Sec. 4. We conclude the paper in Sec. 5.

2 Security Definition

2.1 Notations

Let a be a real number. We denote by $\lfloor a \rfloor$ the largest integer $b \leq a$, by $\lceil a \rceil$ the smallest integer $b \geq a$, and by $\lceil a \rfloor$ the largest integer $b \leq a + 1/2$. For positive real numbers a and b, let $[a]$ denote the set $\{0, 1 \ldots, \lfloor a \rfloor - 1\}$ and $[a, b]$ the set $\{\lfloor a \rfloor, \ldots, \lfloor b \rfloor\}$ and $[-a, b]$ denote the set $\{-\lfloor a \rfloor, \ldots, \lfloor b \rfloor\}$.

By $\mathrm{neg}(\lambda)$ we denote a negligible function, i.e., a function f such that $f(\lambda) < 1/p(\lambda)$ holds for all polynomials $p(\lambda)$ and all sufficiently large λ.

We also use the shorthand notation $\{\mathsf{PK}\}_N$ and $\{\mathsf{SK}\}_N$, $N \in \mathbb{N}$, to mean the sets $\{\mathsf{PK}_1, \ldots, \mathsf{PK}_N\}$ and $\{\mathsf{SK}_1, \ldots, \mathsf{SK}_N\}$ respectively.

2.2 A High Level Description

Before giving a formal definition of verifiable ring encryption, we begin with a high level discussion of this notion in order to let readers understand more easily.

We start by the description of an ordinary verifiable encryption. A verifiable encryption scheme proves that a ciphertext encrypts a plaintext satisfying a certain relation $\mathcal{R}$. The relation $\mathcal{R}$ is defined by a generator algorithm $\mathcal{G}'$ which on input a security parameter λ outputs a binary relation $W \times \Delta$. For $\delta \in \Delta$, an element $w \in W$ such that $(w, \delta) \in \mathcal{R}$ is called a *witness* for δ. The encryptor will be given a value δ, a witness w for δ, then encrypts w to generate a ciphertext

ψ. Later, the encryptor may prove to another party that ψ decrypts to a witness for δ. In this system, the honest verifier will output accept or reject. If the system is sound, the verifier accepts a proof means that with overwhelming probability the ciphertext ψ can be decrypted to a witness for δ.

We extend this concept into a group of N designated receivers. In a verifiable ring encryption scheme, a prover proves that a ciphertext encrypts a plaintext satisfying one of the certain relation $\mathcal{R}$ which is corresponding to one of the receiver. The idea is that the encryptor will be given a value w, which is a witness for δ where $(w, \delta) \in \mathcal{R}$, and randomly generates other $N - 1$ witnesses and the corresponding group elements.

Note that for an interactive proof system, both the prover and the verifier are required to interact in order to have the verifier convinced. If the proof system is non-interactive, the proof is carried out in a non-interactive fashion – the prover (or the encryptor) generates a proof transcript that can be used to convince a verifier at any later time that one (out of N) of the receivers can decrypt the corresponding witness of that group element δ. However, the verifier still cannot compute the identity of the actual decryptor.

2.3 Defining Verifiable Ring Encryption

A *Verifiable Ring Encryption* (VRE) scheme is actually a group encryption scheme with add-on *Verifiability*. A group encryption scheme is a generalization of a public key encryption scheme. Entities involved in such a scheme include an *encryptor* and a group of N *users*. The encryptor has a secret message m which he wants to send to a certain designated one out of the N users in the group, so that the secret message can be decrypted only by the designated member. In other words, a VRE scheme, apart from allowing a secret message to be encrypted to some designated members, provides with the encryptor the ability to prove that a ciphertext encrypts a plaintext satisfying certain relation $\mathcal{R}$.

The relation $\mathcal{R}$ is defined by a generator algorithm $\mathcal{G}'$ which on input 1^λ outputs a description $\Psi = \Psi[\mathcal{R}, W, \Delta]$ of a binary relation $\mathcal{R}$ on $W \times \Delta$. We require that the sets $\mathcal{R}$, W, and Δ are easy to recognize (given Ψ). For $\delta \in \Delta$, an element $w \in W$ such that $(w, \delta) \in \mathcal{R}$ is called a witness for δ. The idea is that the encryptor will be given a value δ, a witness w for δ, and a label L, and then encrypts w under L, yielding a ciphertext ψ. After this, the encryptor may prove to another party that ψ decrypts under L to a witness for δ. In carrying out the proof, the encryptor will need to make use of the random coins that were used by the encryption algorithm.

Now, a Ver-Gp-Enc scheme is a tuple of $(\mathcal{S}, \mathcal{G}, \mathcal{E}, \mathcal{D}, \mathcal{P}, \mathcal{V})$ defined as follows:

- $\mathsf{param} \leftarrow \mathcal{S}(1^\lambda)$, the probabilistic polytime (PPT) *Setup* algorithm that on input security parameter $1^\lambda, \lambda \in \mathbb{N}$, outputs and publishes a set of system's parameters param that also includes the security parameter 1^λ, and a description $\Psi[\mathcal{R}, W, \Delta] \leftarrow \mathcal{G}'(1^\lambda)$.
- $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathcal{G}(\mathsf{param}, 1^{\lambda_i})$, the PPT *Key Generation* algorithm that on input the set of system's parameters param and security parameter $1^{\lambda_i}, \lambda_i \in \mathbb{N}$,

where $\lambda_i \geq \lambda$, outputs a public-key/private key pair $(\mathsf{PK}_i, \mathsf{SK}_i)$. PK_i includes also the security parameter 1^{λ_i}.

- $\psi \leftarrow \mathcal{E}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L)$, the PPT *Encryption* algorithm that takes as input the set of system's parameters param, the group size $N \in \mathbb{N}$ of size polynomial in λ, a set of N public keys $\{\mathsf{PK}\}_N$, an index $\pi \in [1, N]$, a message $w \in W$ which is the witness of $\delta \in \Delta$, and a label $L \in \{0,1\}^*$, and outputs a ciphertext ψ. We denote by $\mathcal{E}'(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L)$ the pair $(\psi, coins)$, where ψ is the output of $\mathcal{E}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L)$ and $coins$ are the random coins used by $\mathcal{E}$ to compute ψ.
- $m/\perp \leftarrow \mathcal{D}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, \mathsf{SK}_\pi, \psi, L)$, the polynomial-time *Decryption* algorithm that takes as input the set of system's parameters param, the group size $n \in \mathbb{N}$ of size polynomial in λ, a set $\{\mathsf{PK}\}_N$ of N public keys, an index $\pi \in [1, N]$, a private key SK_π, a ciphertext ψ, and a label $L \in \{0,1\}^*$, and outputs either a message $m \in \mathcal{M}$, or a special symbol $\perp$. The output of the algorithm implicitly defines the domain of m, that we denote by $\mathcal{M}$.
- $\mathsf{proof} \leftarrow \mathcal{P}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L, \psi, coins)$, the PPT *Proof* algorithm that takes as input the tuple $(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L, \psi, coins)$ such that $(\psi, coins)$ is the output of some $\mathcal{E}'(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L)$, and outputs a proof proof.
- $0/1 \leftarrow \mathcal{V}(\mathsf{param}, N, \{\mathsf{PK}\}_N, L, \psi, \mathsf{proof})$, the polynomial-time *Verification* algorithm that takes as input the tuple $(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, L, \psi)$ such that ψ is the output of some $\mathcal{E}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L)$ for some $\pi \in [1, N]$, $w \in \mathcal{M}$ and $\delta \in \Delta$, and outputs either 0 or 1, indicating accept or reject respectively.

Here we take a more relaxed approach in order to make it to be more convenient and adequate for practical applications. Instead of requiring the ciphertext to be decrypted to a witness, we only require that a witness can be easily reconstructed from the plaintext using some efficient reconstruction algorithm *recon*. We believe that this definition is more suitable for many applications.

Definition 1. *The above Ver-Gp-Enc scheme is a Verifiable Group Encryption scheme, if it is (1) correct, (2) sound, (3) zero-knowledge and (4) anonymous, as defined in the following.*

Correctness: *A Ver-Gp-Enc is correct if it satisfies both* Verification Correctness *and* Decryption Correctness *defined below.*

- (Verification Correctness.) *For all* $\mathsf{param} \leftarrow \mathcal{S}(1^\lambda)$*, for all* $N \in \mathbb{N}$ *of size polynomial in* λ*, for all* $\lambda_i \geq \lambda$*,* $i \in [1, N]$*, for all* $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathcal{G}(\mathsf{param}, 1^{\lambda_i})$*,* $i \in [1, N]$*, for all* $(w, \delta) \in \mathcal{R}$*, for all* $L \in \{0,1\}^*$*, for all* $\pi \in [1, N]$*, for all* $(\psi, coins) \leftarrow \mathcal{E}'(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L)$*, for all*

$$\mathsf{proof} \leftarrow \mathcal{P}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L, \psi, coins),$$

$$\Pr[x \leftarrow \mathcal{V}(\mathsf{param}, N, \{\mathsf{PK}\}_N, L, \psi, \mathsf{proof}) : x = 1] = 1 - \mathrm{neg}(\lambda).$$

– (Decryption Correctness.) *For all* $\mathsf{param} \leftarrow \mathcal{S}(1^\lambda)$*, for all* $N \in \mathbb{N}$ *of size polynomial in* λ*, for all* $\lambda_i \geq \lambda$*,* $i \in [1, N]$*, for all* $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathcal{G}(\mathsf{param}, 1^{\lambda_i})$*,* $i \in [1, N]$*, for all* $\pi \in [1, N]$*, for all* $w \in \mathcal{M}$*, for all* $L \in \{0,1\}^*$*, for all*

$$\psi \leftarrow \mathcal{E}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L),$$

$$\Pr[\tilde{m} \leftarrow \mathcal{D}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, \mathsf{SK}_\pi, \psi, L) : m = \tilde{m}] = 1 - \mathrm{neg}(\lambda).$$

Soundness: *For all PPT adversaries* $\mathcal{A}_1, \mathcal{A}_2$*, and some reconstruction PPT algorithm recon,*

$$\begin{aligned}
&\Pr[\ \mathsf{param} \leftarrow \mathcal{S}(1^\lambda);\\
&\quad (N, \lambda_1, \ldots, \lambda_N) \leftarrow \mathcal{A}_1(\mathsf{param}),\\
&\quad \textit{where } N \textit{ has a size polynomial in } \lambda \textit{ and } \lambda_i \geq \lambda \textit{ for all } i \in [1, N];\\
&\quad (\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathcal{G}(\mathsf{param}, 1^{\lambda_i}),\ \textit{for all } i \in [1, N];\\
&\quad (\delta, \psi, L, \mathsf{proof}) \leftarrow \mathcal{A}_2(\mathsf{param}, N, \{\mathsf{PK}\}_N, \{\mathsf{SK}\}_N);\\
&\quad x \leftarrow \mathcal{V}(\mathsf{param}, N, \{\mathsf{PK}\}_N, L, \psi, \mathsf{proof});\\
&\quad m_j \leftarrow \mathcal{D}(\mathsf{param}, N, \{\mathsf{PK}\}_N, j, \mathsf{SK}_j, \psi, L\}),\ \textit{for all } j \in [1, N];\\
&\quad w_j \leftarrow \mathit{recon}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \delta, m_j),\ \textit{for all } j \in [1, N] :\\
&\quad x = 1 \wedge (\forall j \in [1, N])((w_j, \delta) \notin \mathcal{R}) \qquad\qquad]\\
&= \mathrm{neg}(\lambda).
\end{aligned}$$

Simply speaking, the definition of soundness above means that if a ciphertext is verified by a verifier to be valid, then there exists one user who can decrypt the ciphertext to the witness of δ, with overwhelming probability.

Zero knowledge: *There exists a PPT simulator Sim such that for all PPT adversaries* $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$*, we have*

$$\begin{aligned}
&\Pr[\ \mathsf{param} \leftarrow \mathcal{S}(1^\lambda);\\
&\quad (N, \lambda_1, \ldots, \lambda_N) \leftarrow \mathcal{A}_1(\mathsf{param}),\\
&\quad \textit{where } N \textit{ has a size polynomial in } \lambda \textit{ and } \lambda_i \geq \lambda \textit{ for all } i \in [1, N];\\
&\quad (\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathcal{G}(\mathsf{param}, 1^{\lambda_i}),\ \textit{for all } i \in [1, N];\\
&\quad (w, \delta, L, \pi) \leftarrow \mathcal{A}_2(\mathsf{param}, N, \{\mathsf{PK}\}_N, \{\mathsf{SK}\}_N),\\
&\quad \textit{where } (w, \delta) \in \mathcal{R},\ L \in \{0,1\}^* \textit{ and } \pi \in [1, N];\\
&\quad (\psi, \mathit{coins}) \leftarrow \mathcal{E}'(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L);\\
&\quad b \leftarrow \{0,1\};\\
&\quad \textit{if } b = 0\\
&\qquad \textit{then } \mathsf{proof} \leftarrow \mathcal{P}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, w, \delta, L, \psi, \mathit{coins})\\
&\qquad \textit{else } \ \mathsf{proof} \leftarrow \mathit{Sim}(\mathsf{param}, N, \{PK\}_N, \delta, \psi, L);\\
&\quad \hat{b} \leftarrow \mathcal{A}_3(\mathsf{param}, N, \{\mathsf{PK}\}_N, \{\mathsf{SK}\}_N, w, \delta, L, \pi, \psi, \mathsf{proof}) :\\
&\quad b = \hat{b} \qquad\qquad]\\
&= 1/2 + \mathrm{neg}(\lambda).
\end{aligned}$$

The definition above means that an adversary cannot distinguish a simulated proof from a proof generated from real execution of algorithms. In other words, the proof is zero-knowledge to a verifier.

Anonymity: *For all PPT adversaries* $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$,

$$
\begin{aligned}
&\Pr[\ \mathit{param} \leftarrow \mathcal{S}(1^\lambda);\\
&\quad (N, \lambda_1, \ldots, \lambda_N) \leftarrow \mathcal{A}_1(\mathit{param}, \Psi),\\
&\quad \textit{where } N \textit{ has a size polynomial in } \lambda \textit{ and } \lambda_i \geq \lambda \textit{ for all } i \in [1, N];\\
&\quad (\mathit{PK}_i, \mathit{SK}_i) \leftarrow \mathcal{G}(\mathit{param}, 1^{\lambda_i}),\ \textit{for all } i \in [1, N];\\
&\quad (w, \delta, L, \pi_0, \pi_1) \leftarrow \mathcal{A}_2(\mathit{param}, N, \{\mathit{PK}\}_N),\\
&\quad \textit{where } (w, \delta) \in \mathcal{R} \textit{ and } \pi_0, \pi_1 \in [1, N] \textit{ are distinct};\\
&\quad b \leftarrow \{0, 1\};\\
&\quad (\psi, coins) \leftarrow \mathcal{E}'(\mathit{param}, N, \{\mathit{PK}\}_N, \pi_b, w, \delta, L);\\
&\quad \mathit{proof} \leftarrow \mathcal{P}(\mathit{param}, N, \{\mathit{PK}\}_N, \pi_b, w, L, \psi, coins);\\
&\quad \hat{b} \leftarrow \mathcal{A}_3(\mathit{param}, N, \{\mathit{PK}\}_N, w, \delta, L, \pi_0, \pi_1, \{\mathit{SK}_i | i \in [1, n] \backslash \{\pi_0, \pi_1\}\}, \psi, \mathit{proof});\\
&\quad \hat{b} = b\ ]\\
&= 1/2 + \mathrm{neg}(\lambda).
\end{aligned}
$$

The definition of anonymity above means that an adversary cannot decide better than random guessing, given a ciphertext together with a corresponding proof transcript, who among the 2 possible designated members is actually designated, even he has corrupted all of the other $(N-2)$ members.

3 The Proposed Scheme

3.1 Key Generation

For each user, select two random ℓ-bit Sophie Germain primes p' and q', with $p' \neq q'$, and compute $p = (2p'+1), q = (2q'+1)$ and $n = pq$, where $\ell = \ell(\lambda)$ is a security parameter which is a polynomial in λ. Choose random $x_1, x_2, x_3 \in_R [n^2/4]$, choose a random $g' \in_R \mathbb{Z}^*_{n^2}$, and compute $g = (g')^{2n}$, $y_1 = g^{x_1}$, $y_2 = g^{x_2}$ and $y_3 = g^{x_3}$.

Let Γ be a cyclic group of order ρ generated by γ. We assume ρ and γ are publicly known, and that ρ is prime. Let $W = [\rho]$ and $\Delta = \Gamma$, and let $\mathcal{R} = \{(w, \delta) \in W \times \Delta : \gamma^w = \delta\}$.

Choose two other $\mathfrak{l}$-bit primes $\mathfrak{p}', \mathfrak{q}'$ and compute $\mathfrak{p} = 2\mathfrak{p}'+1$, $\mathfrak{q} = 2\mathfrak{q}'+1$ and $\mathfrak{n} = \mathfrak{p}\mathfrak{q}$, and choose $\mathfrak{g}, \mathfrak{h}$ as two generators of $\mathfrak{G}_{\mathfrak{n}'} \subset \mathbb{Z}^*_{\mathfrak{n}}$, where $\mathfrak{n}' = \mathfrak{p}'\mathfrak{q}'$ and $\mathfrak{G}_{\mathfrak{n}'}$ is the subgroup of $\mathbb{Z}^*_{\mathfrak{n}}$ of order $\mathfrak{n}'$, and $\mathfrak{l} = \mathfrak{l}(\lambda)$ which is a polynomial in λ.

The public key of this user is $(n, g, y_1, y_2, y_3, \mathfrak{n}, \mathfrak{g}, \mathfrak{h}, h, \rho, \gamma)$ and the secret key is (x_1, x_2, x_3, p, q) where $h = (1 + n \bmod n^2) \in \mathbb{Z}^*_{n^2}$. We further define $H : \{0,1\}^* \rightarrow \{0,1\}^k$ be a collision resistant hash function and $\mathsf{abs} : \mathbb{Z}^*_{n^2} \rightarrow \mathbb{Z}^*_{n^2}$ maps $(a \bmod n^2)$, where $0 < a < n^2$, to $(n^2 - a \bmod n^2)$ if $a > n^2/2$, and to $(a \bmod n^2)$, otherwise.

For a list of N users, we denote PK_i, the public key of user i be $(n_i, g_i, y_{1_i}, y_{2_i}, y_{3_i}, \mathfrak{n}_i, \mathfrak{g}_i, \mathfrak{h}_i, h_i, \rho_i, \gamma_i)$ and the corresponding secret key SK_i is $(x_{1_i}, x_{2_i}, x_{2_i}, p_i, q_i)$. For simplicity, we let L denote the list of the public keys of N users.

3.2 Encryption and Ciphertext Validity Proof

The prover sends an encrypted message to one of the N receivers such that only one of them can decrypt the message. At the same time, any verifier having the

public keys of those N receivers can verify that the ciphertext can be decrypted by at least one of the receivers yet does not know the identity of this targeted receiver.

We use a special kind of encryption scheme by Camenisch and Shoup [8] where the plaintext is the discrete log of a group element. Then we apply a 1-out-of-n proof-of-knowledge methodology to achieve our goal.

To encrypt a message $m \in [n_\pi]$ under L, the list of public keys of N users, the prover executes the following algorithm:

1. For $i = 1, \ldots, N, i \neq \pi$, randomly generate $m_i \in_R [n_i]$ and compute $\delta_i = \gamma_i^{m_i}$. For π, compute $\delta_\pi = \gamma_\pi^m$.
2. Randomly generate $r_\pi, s_\pi \in_R [n_\pi/4]$ and compute $u_\pi = g_\pi^{r_\pi},\ e_\pi = y_{1_\pi}^{r_\pi} h_\pi^m,\ v_\pi = \mathsf{abs}((y_{2_\pi} y_{3_\pi}^{H(u_\pi, e_\pi)})^{r_\pi}),\ \mathfrak{t}_\pi = \mathfrak{g}_\pi^m \mathfrak{h}_\pi^{s_\pi}$
3. Randomly generate $r'_\pi \in_R [-n_\pi 2^{k+k'-2},\ n_\pi 2^{k+k'-2}],\ s'_\pi \in_R [-\mathfrak{n}_\pi 2^{k+k'-2}, \mathfrak{n}_\pi 2^{k+k'-2}],\ m'_\pi \in_R [-\rho_\pi 2^{k+k'-2}, \rho_\pi 2^{k+k'-2}]$ and compute $u'_\pi = g_\pi^{2r'_\pi},\ e'_\pi = y_{1_\pi}^{2r'_\pi} h_\pi^{2m'_\pi},\ v'_\pi = (y_{2_\pi} y_{3_\pi}^{H(u_\pi, e_\pi)})^{2r'_\pi},\ \delta'_\pi = \gamma_\pi^{m'_\pi},\ \mathfrak{t}'_\pi = \mathfrak{g}_\pi^{m'_\pi} \mathfrak{h}_\pi^{s'_\pi}$ and $c_{\pi+1} = H(L, \delta_\pi, u'_\pi, e'_\pi, v'_\pi, \delta'_\pi, \mathfrak{t}'_\pi)$.
4. For $i = \pi + 1, \ldots, n, 1, \ldots, \pi - 1$, randomly generate $\tilde{r}_i \in_R [-n_i 2^{k+k'-2}, n_i 2^{k+k'-2}],\ \tilde{s}_i \in_R [-\mathfrak{n}_i 2^{k+k'-2}, \mathfrak{n}_i 2^{k+k'-2}],\ \tilde{m}_i \in_R [-\rho_i 2^{k+k'-2}, \rho_i 2^{k+k'-2}]$, $u_i, e_i, v_i \in_R \mathbb{Z}_{n^2}^*,\ \mathfrak{t}_i \in_R \mathbb{Z}_{\mathfrak{n}^2}^*$ and compute $u'_i = u_i^{2c_i} g_i^{2\tilde{r}_i},\ e'_i = e_i^{2c_i} y_{1_i}^{2\tilde{r}_i} h_i^{2\tilde{m}_i}$ $v'_i = v_i^{2c_i} (y_{2_i} y_{3_i}^{H(u_i, e_i)})^{2\tilde{r}_i} \delta'_i = \delta_i^{c_i} \gamma_i^{\tilde{m}_i},\ \mathfrak{t}'_i = \mathfrak{t}_i^{c_i} \mathfrak{g}_i^{\tilde{m}_i} \mathfrak{h}_i^{\tilde{s}_i},\ c_{i+1} = H(L, \delta_i, u'_i, e'_i, v'_i, \delta'_i, \mathfrak{t}'_i)$
5. Compute $\tilde{r}_\pi = r'_\pi - c_\pi r_\pi,\ \tilde{s}_\pi = s'_\pi - c_\pi s_\pi,\ \tilde{m}_\pi = m'_\pi - c_\pi m_\pi$ (all are computed in $\mathbb{Z}$)
6. Output the ciphertext ψ and the proof proof, where

$$\psi := ((u_1, e_1, v_1), \ldots, (u_N, e_N, v_N)), \text{ and}$$

$$\mathsf{proof} := ((\delta_1, \mathfrak{t}_1, \tilde{r}_1, \tilde{s}_1, \tilde{m}_1), \ldots, (\delta_N, \mathfrak{t}_N, \tilde{r}_N, \tilde{s}_N, \tilde{m}_N), c_1).$$

Note that we describe the *Encryption* algorithm and the *Proof* algorithm in a combined fashion to allow a neat presentation. It should also be a common practice to do both in one shot in real applications. However, they can always be done separately if desired.

3.3 Decryption

Assume user π is the actual decryptor. To decrypt a ciphertext ψ using his own secret key SK_π, user π check whether $\mathsf{abs}(v_\pi) \stackrel{?}{=} v_\pi$ and $u_\pi^{2(x_{2_\pi} + H(u_\pi, e_\pi) x_{3_\pi}} \stackrel{?}{=} v_\pi^2$. If this does not hold, then output reject and halt. Next, let $t_\pi = 2^{-1} \bmod n_\pi$ and compute $\bar{m} = (e_\pi / u_\pi^{x_{1_\pi}})^{2t_\pi}$. If $\bar{m}$ is of the form h_π^m for some $m \in [n_\pi]$, then output m. Otherwise, output reject.

3.4 Verification

Any verifier on input L, the list of public keys of those N users, and the ciphertext ψ, can verify that ψ can be decrypted by at least one of the N users. That is, at least one user π can reconstruct the plaintext, which is the discrete log of the group element δ_π. Yet the verifier cannot compute the identity of this actual decryptor and cannot compute the plaintext.

The verification algorithm is as follows.

1. For $i = 1, \ldots, N$, compute $\hat{u}_i = u_i^{2c_i} g_i^{2\tilde{r}_i}$, $\hat{e}_i = e_i^{2c_i} y_{1_i}^{2\tilde{r}_i} h_i^{2\tilde{m}_i}$, $\hat{v}_i = v_i^{2c_i}(y_{2_i} y_{3_i}^{H(u_i,e_i)})^{2\tilde{r}_i}$, $\hat{\delta}_i = \delta_i^{c_i} \gamma_i^{\tilde{m}_i}$, $\hat{\mathfrak{t}}_i = \mathfrak{t}_i^{c_i} \mathfrak{g}_i^{\tilde{m}_i} \mathfrak{h}_i^{\tilde{s}_i}$ and $c_{i+1} = H(L, \delta_i, \hat{u}_i, \hat{e}_i, \hat{v}_i, \hat{\delta}_i, \hat{\mathfrak{t}}_i)$ if $i \neq n$
2. Check whether
$$c_1 \stackrel{?}{=} H(L, \delta_N, \hat{u}_N, \hat{e}_N, \hat{v}_N, \hat{\delta}_N, \hat{\mathfrak{t}}_N)$$
If yes, output accept. Otherwise, output reject.

4 Security Analysis

The assumptions used for proving our scheme are the following.

Assumption 1 (Strong RSA Assumption). *Given a composite modulus n and a random element $g \in \mathbb{Z}_n^*$, it is hard to compute $h \in \mathbb{Z}_n^*$ and integer $e > 1$ such that $h^e = g$.*

Assumption 2. (Paillier Decision Composite Residuosity (DCR) Assumption [13]) *Given only n, it is hard to distinguish random elements of $\mathbb{Z}_{n^2}^*$ from random elements of the subgroup of $\mathbb{Z}_{n^2}^*$ consisting of all n-th powers of elements in $\mathbb{Z}_{n^2}^*$.*

To be complete, one needs to specify more precisely the distribution from which n is drawn. We specify that n is of the form pq, where $p = 2p' + 1, q = 2q' + 1$, and p' and q' are uniformly distributed over all ℓ-bit numbers such that p, q, p', q' are prime and $p' \neq q'$, where ℓ is the security parameter.

Theorem 1. *Under the strong RSA and DCR assumption, our proposed scheme is a Verifiable Ring Encryption scheme in the random oracle model.*

The proof can be found in Appendix A.

5 Conclusion

In this paper, we propose a *Verifiable Ring Encryption* scheme for ad hoc groups. Different from previous verifiable encryption schemes which are for one designated receiver, our proposed scheme is targeted for a group of N receivers. However, only one of them is able to decrypt the ciphertext while others cannot. Any public verifier (who has the public keys of those N users) can verify this

fact yet he cannot compute the identity of this actual decryptor. We propose a concrete construction and prove its security in the random oracle model. We believe this kind of schemes will attract many applications in practice.

In addition, there are open problems left such as constructing a verifiable ring encryption scheme that supports partial or fully separability [9,6]. To build up a verifiable subgroup encryption, that is, a targeted subgroup of t members out of a group of N members are able to decrypt the message anonymously, is another interesting future research. The physical size of the ciphertext of our proposed scheme grows linearly with the number of designated receivers. It is another open problem to make the size of the ciphertext to be irrelevant to the group size.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Proc. EUROCRYPT 98*, pages 591–606. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1403.
2. F. Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *Proc. Smart Card Research and Applications (CARDIS) 1998*, pages 213–220. Springer-Verlag, 2000. Lecture Notes in Computer Science No. 1820.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
4. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *Proc. ASIACRYPT 2000*, pages 331–345. Springer-Verlag, 2000. Lecture Notes in Computer Science No. 1976.
5. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocations. In *Proc. EUROCRYPT 2001*, pages 93–118. Springer-Verlag, 2001. Lecture Notes in Computer Science No. 2045.
6. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *Proc. CRYPTO 99*, pages 413–430. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1666.
7. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. http://eprint.iacr.org/2002/161/, 2002.
8. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proc. CRYPTO 2003*, pages 126–144. Springer-Verlag, 2003. Leture Notes in Computer Science No. 2729.
9. J. Kilian and E. Petrank. Identity escrow. In *Proc. CRYPTO 98*, pages 169–185. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1642.
10. J. Liu, V. Wei, and D. Wong. Custodian-hiding verifiable encryption. In *WISA 2004*, pages 54–67. Springer-Verlag, 2004. Lecture Notes in Computer Science No. 3325.
11. J. Liu, V. Wei, and D. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *ACISP04*, pages 325–335. Springer-Verlag, 2004. Lecture Notes in Computer Science No. 3108.

12. K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In *Proc. CRYPTO 98*, pages 354–369. Springer-Verlag, 1998. LNCS Vol. 1462.
13. P. Paillier. Public-key cryptosystems based on composite residuosity classes. In *Proc. EUROCRYPT 99*, pages 223–239. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1592.
14. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proc. EUROCRYPT 96*, pages 387–398. Springer-Verlag, 1996. LNCS Vol. 1070.
15. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proc. ASIACRYPT 2001*, pages 552–565. Springer-Verlag, 2001. Lecture Notes in Computer Science No. 2248.
16. M. Stadler. Publicly verifiable secret sharing. In *Proc. EUROCRYPT 96*, pages 191–199. Springer-Verlag, 1996. Lecture Notes in Computer Science No. 1070.

A Proof of Theorem 1

Proof. Correctness of our proposed scheme is trivial and its proof is thus omitted. The proof of the theorem is then a direct implication of the three lemmas that follow. □

Lemma 1 (Soundness). *Our proposed scheme is sound in the random oracle model if the Strong RSA assumption holds.*

Proof. Assume there is a PPT algorithm $\mathcal{P}^*$, which can produce a ciphertext ψ (corresponding to δ) with non-negligible probability such that $\mathcal{V}$ outputs accept but no one can decrypt, or compute m' such that $(m', \delta) \in \mathcal{R}$. That is,

$$\Pr[(\mathcal{D}(\mathsf{param}, N, \{\mathsf{PK}\}_N, \pi, \mathsf{SK}_\pi, \psi, L), \delta) \notin \mathcal{R}] > \mathsf{neg}(\lambda)$$

for at least one $\pi \in \{1, \ldots, N\}$. for some PPT algorithm $\mathcal{V}$ and $\mathcal{D}$.

We construct a PPT simulator (the reduction master) $\mathcal{M}$ which has N private keys $\mathsf{SK}_1, \ldots, \mathsf{SK}_N$ and calls $\mathcal{P}^*$ to compute an integer x such that $(x, \delta) \in \mathcal{R}$.

$\mathcal{M}$ also controls the random oracle H. It flips coins for H and records queries to the oracle. It maintains the consistnecy of H. $\mathcal{P}^*$ is allowed the query the random oracle at most q_H times.

$\mathcal{P}^*$ generates a ciphertext ψ (corresponding to δ), consists of c_1, $(u_1, e_1, v_1, \mathfrak{t}_1, \tilde{r}_1, \tilde{s}_1, \tilde{m}_1)$, $\ldots$, $(u_N, e_N, v_N, \mathfrak{t}_N, \tilde{r}_N, \tilde{s}_N, \tilde{m}_N)$ where it satisfies the verification including the following N equations: $c_{i+1} = H(L, \delta_i, \hat{u}_i, \hat{e}_i, \hat{v}_i, \hat{\delta}_i, \hat{\mathfrak{t}}_i)$ for $i = 1, \ldots, N-1$ and $c_1 = H(L, \delta_N, \hat{u}_N, \hat{e}_N, \hat{v}_N, \hat{\delta}_N, \hat{\mathfrak{t}}_N)$ where $\hat{u}_i = u_i^{2c_i} g_i^{2\tilde{r}_i}$, $\hat{e}_i = e_i^{2c_i} y_{1_i}^{2\tilde{r}_i} h_i^{2\tilde{m}_i}$, $\hat{v}_i = v_i^{2c_i} (y_{2_i} y_{3_i}^{H(u_i, e_i)})^{2\tilde{r}_i}$ $\hat{\delta}_i = \delta_i^{c_i} \gamma_i^{\tilde{m}_i}$, $\hat{\mathfrak{t}}_i = \mathfrak{t}_i^{c_i} \mathfrak{g}_i^{\tilde{m}_i} \mathfrak{h}_i^{\tilde{s}_i}$ for $i = 1, \ldots, N$

The master $\mathcal{M}$ will invoke $\mathcal{A}$ with constructed inputs, receive and process outputs from $\mathcal{A}$, and may invoke $\mathcal{P}^*$ for multiple times depending on $\mathcal{P}^*$'s outputs from previous invocations. In the random oracle model, $\mathcal{M}$ flips the coins for the random oracle H record queries to the oracle. Consider each invocation of $\mathcal{P}^*$ to be recorded on a simulation transcript tape. Some transcripts produce successful ciphertext. Others do not.

Let $\mathbf{E}$ be the event that each of the N queries corresponding to the N Verification queries have been included in the q_H queries $\mathcal{P}^*$ made to the random oracles. In the event $\bar{\mathbf{E}}$, $\mathcal{M}$ needs to flip additional coins in order to verify $\mathcal{P}^*$'s ciphertext. Then the probability of c_1 satisfying the (final) Verification equation is at most $1/(2^k - q_H)$ because $\mathcal{P}^*$ can only guess the outcomes of queries used in Verification that he has not made. Therefore

$$\begin{aligned}\mathsf{neg}(\lambda) &< \Pr[\mathbf{E}]\Pr[\mathcal{P}^* \text{ succeed}|\mathbf{E}] + \Pr[\bar{\mathbf{E}}]\Pr[\mathcal{P}^* \text{ succeed}|\bar{\mathbf{E}}] \\ &\leq \Pr[\mathbf{E}]\Pr[\mathcal{P}^* \text{ succeed}|\mathbf{E}] + 1 \cdot (\frac{1}{2^k - q_H})\end{aligned}$$

and

$$\Pr[\mathbf{E} \text{ and } \mathcal{P}^* \text{ succeed}] > \mathsf{neg}(\lambda) - (\frac{1}{2^k - q_H})$$

Hence the probability of $\mathcal{P}^*$ returning a valid ciphertext and having already queried the random oracle for all the N queries used in Verification is essentially greater than $\mathsf{neg}(\lambda)$ as $\frac{1}{2^k - q_H}$ is negligibly small.

Therefore, in each $\mathcal{P}^*$ transcript which produced a valid ciphertext, there exists N queries to H, denoted by $X_{i_1}, \cdots, X_{i_N}$, $1 \leq i_1 < \cdots < i_N$, such that they match the N queries made in Verification. This happens with each transcript that $\mathcal{P}^*$ successfully produces a valid ciphertext, with negligible exceptions.

In creating a successful ciphertext ψ by $\mathcal{P}^*$, consider the set of all queries made by $\mathcal{P}^*$ that were used (including duplicate queries) in Verification. Let X_{i_1}, $\cdots$, X_{i_N} denote the first appearance of each of the queries used in Verification, $i_1 < \cdots < i_N$. Let π be such that $X_{i_N} := H(L, \delta_{\pi-1}, \hat{u}_{\pi-1}, \hat{e}_{\pi-1}, \hat{v}_{\pi-1}, \hat{\delta}_{\pi-1}, \hat{\mathfrak{t}}_{\pi-1})$ in Verification. We call π the *gap* of ψ.

We call a successful creation of ψ by $\mathcal{P}^*$ a (ℓ, π)-ψ if $i_1 = \ell$. That is, the first appearance of all Verification-related queries is the ℓ-th query and the gap equals π. There exist ℓ and π, $1 \leq \ell \leq q_H$, $1 \leq \pi \leq N$, such that the probability $\mathcal{P}^*$ produces (ℓ, π)-ψ is no less than $1/(Nq_H\mathsf{neg}(\lambda))$.

In the following, $\mathcal{M}$ will do a rewind-simulation for each value of ℓ and π.

In the rewind-simulation for a given (ℓ, π), $\mathcal{M}$ first invokes $\mathcal{P}^*$ to obtain its output and its Turing transcript $\mathcal{T}$. $\mathcal{M}$ computes the output and the transcript to determine whether they form a successful (ℓ, π)-ψ. If not, abort. Otherwise continue. This can be done in at most polynomial time because $\mathcal{M}$ records queries made by $\mathcal{P}^*$ to the random oracles. The transcript $\mathcal{T}$ is rewound to the ℓ-th query and given to $\mathcal{P}^*$ for a rewind-simulation to generate transcript $\mathcal{T}'$. New coin flips independent of those in $\mathcal{T}$ are made for all queries subsequent to the ℓ-th query while maintaining consistencies with the prior queries. $\mathcal{T}$ and $\mathcal{T}'$ use the same code in $\mathcal{P}^*$. The ℓ-th query, common to $\mathcal{T}$ and $\mathcal{T}'$, is denoted $c_{\pi+1} = H(L, \delta_\pi, u'_\pi, e'_\pi, v'_\pi, \gamma^u_\pi, \mathfrak{t}'_\pi)$ $\mathcal{M}$ knows γ^u_π but not u at the time of the rewind. After $\mathcal{P}^*$ returns the output from the rewind simulation, $\mathcal{M}$ proceeds to compute the DL of δ_π, that is, u.

By the forking lemma [14], heavy-row lemma [12] or the Rewind-on-Success lemma [11], there exists non-negligible probability that $\mathcal{P}^*$ produces two (ℓ, π)-ψ

from the tape $\mathcal{T}$ and a rewind-simulation tape $\mathcal{T}'$ with $\gamma_\pi^u = \gamma_\pi^{\tilde{m}_\pi + c_\pi m}$ from $\mathcal{T}$ and $\gamma_\pi^u = \gamma_\pi^{\tilde{m'}_\pi + c'_\pi m}$ from $\mathcal{T}'$

Solve for the equations to obtain m. Using the argument in the proof of Theorem 4 in [7] (and by the Strong RSA Assumption), $(m, \delta) \in \mathcal{R}$. That is, ψ is an encryption of the witness of δ. Desired contradition occurs.

Lemma 2 (Zero-knowledge). *Our proposed scheme is zero-knowledge in the random oracle model.*

Proof. (Sketch.) This is rather obvious due to the symmetry enjoyed by the ring-structure of the ciphertext validity proof. □

Lemma 3 (Anonymity). *Our proposed scheme is anonymous in the random oracle model if DCR assumption holds.*

Proof. (Sketch.) Observe that $(u_i, e_i, v_i, \mathfrak{t}_i, \tilde{r}_i, \tilde{s}_i, \tilde{m}_i), i = 1, \ldots, n, i \neq \pi$ are all random numbers chosen uniformly. At the closing point, $(u_\pi, e_\pi, v_\pi, \mathfrak{t}_\pi, \tilde{r}_\pi, \tilde{s}_\pi, \tilde{m}_\pi)$ also distribute uniformly since r_π and s_π are uniformly chosen from $[n_\pi/4]$. Remaining c_1 is the output of a hash function which can be regarded as a random number as well. □