Non-Slanderability of Linkable Spontaneous Anonymous Group Signature (LSAG)

Veronika Kuchta[1] and Joseph K. Liu[2]

[1] University of Queensland, Australia
[2] Monash University, Australia

Abstract. In this paper, we formally prove the non-slanderability property of the first linkable ring signature paper in ACISP 2004 (in which the notion was called linkable spontaneous anonymous group signature (LSAG)). The rigorous security analysis will give confidence to any future construction of Ring Confidential Transaction (RingCT) protocol for blockchain systems which may use this signature scheme as the basis.

1 Linkable Ring Signature and Non-Slanderability

The concept of Linkable Ring Signature was first proposed in Liu *et al.* [2] as the notion of *Linkable Spontaneous Anonymous Group Signature* (LSAG) which was later renamed as Linkable Ring Signature in [3]. Different from a normal ring signature, a linkable ring signature allows any verifier to detect if two signatures are generated by the same signer or not. But the anonymity remains preserved either computationally or unconditionally. It is found useful in the constrcution of Ring Confidential Transaction (RingCT) [4] protocol that provides user privacy for blockchain transactions: no one knows who are the sender and the receiver of the transaction, and the transaction amount is also hidden. RingCT is used in Monero, which is the largest privacy-preserving blockchain cryptocurrency in terms of market capitalization.

In the first appearance of linkable ring signature (under the name of LSAG) in [2], only three security notions were considered: Unforgeability, Anonymity and Linkability. Unforgeability and anonymity were defined as the same as normal ring signature scheme, while linkability means no one can generate two signatures being unlinked.

Later on another notion called *Non-Slanderability* has also been considered in linkable ring signature. Non-slanderability means no one can slander another honest user for being linked. In other words, no one can generate a signature which is linked to another signature generated by a different signer. Non-Slanderability first appeared in [3]. The same security requirement falls under the notion of linkability as a sub-requirement in [8].

Non-Slanderability is considered as a necessary security requirement for linkable ring signature if it is deployed in RingCT construction [5,7,1,6,9]. However, the non-slanderability property is never proven for the first construction of linkable ring signature (LSAG) in [2]. In this paper, we formally give a security proof

of the Non-Slanderability for [2]. This provides confidence for future construction of RingCT if they choose to use [2] as the basis.

2 LSAG Signature [Liu et al. ACISP2004 [2]]

2.1 Signature Scheme

Key Generation. Let $\mathbb{G} =< g >$ be a group of prime order q such that the underlying discrete logarithm problem is intractable. Let $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ and $H_2 : \{0,1\}^* \to \mathbb{G}$ be some statistically independent cryptographic hash functions. For $i = 1, \ldots, n$, each user i has a distinct public key y_i and a private key x_i such that $y_i = g^{x_i}$. Let $L = \{y_1, \ldots, y_n\}$ be the list of n public keys.

Signature generation. Given message $m \in \{0,1\}^*$, list of public keys $L = \{y_1, \ldots, y_n\}$, private key x_π corresponding to $y_\pi, 1 \leq \pi \leq n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute $c_{\pi+1} = H_1(L, \tilde{y}, m, g^u, h^u)$.
3. For $i = \pi + 1, \ldots, n, 1, \ldots, \pi - 1$, pick $s_i \in_R \mathbb{Z}_q$ and compute $c_{i+1} = H_1(L, \tilde{y}, m, g^{s_i}, y_i^{c_i}, h^{s_i}\tilde{y}^{c_i})$.
4. Compute $s_\pi = u - x_\pi c_\pi \mod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \ldots, s_n, \tilde{y})$. Remarks: Other methods of generating h can be used, provided it is computable by the public verifier and that it does not damage security. We shall see examples in some of our applications.

Signature Verification. A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \ldots, s_n, \tilde{y})$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \ldots, n$, compute $z' = g^{s_i} y^{c_i}$, $z'' = h^{s_i}\tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z_i', z_i'')$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z_n', z_n'')$. If yes, accept. Otherwise, reject.

Linkability. For a fixed list of public keys L, given two signatures associating with L, namely $\sigma_L'(m') = (c', s', \ldots, s', \tilde{y}')$ and $\sigma_L''(m'') = (c'', s'', \ldots, s'', \tilde{y}'')$, where m' and m'' are some messages, a public verifier after verifying the signatures to be valid, checks if $\tilde{y}' = \tilde{y}''$. If the congruence holds, the verifier concludes that the signatures are created by the same signer. Otherwise, the verifier concludes that the signatures are generated by two different signers.

For a valid signature $\sigma_L(m) = (c_1, s_1, \ldots, s_n, \tilde{y})$ on some message m and some list of public keys L, an investigator subpoenas a private key x_i from user i.If x_i is the private key of some $y_i \in L$ (that is, $y_i = g^{x_i}$) and $\tilde{y} = H_2(L)^{x_i}$, then the investigator conducts that the authorship of the signature belongs to user i.

2.2 Nonslanderability Definition

Security of LRS schemes has four aspects: unforgeability, anonymity, linkability, and nonslanderability. Before giving their definition, we consider the following oracles which together model the ability of the adversaries in breaking the security of the schemes:

- $pk_i \leftarrow \mathcal{JO}(\perp)$. The Joining Oracle, on request, adds a new user to the system. It returns the public key $pk \in \mathcal{PK}$ of the new user.
- $sk_i \leftarrow \mathcal{CO}(pk_i)$. The Corruption Oracle, on input a public key $pk_i \in \mathcal{PK}$ that is a query output of $\mathcal{JO}$, returns the corresponding private key $sk_i \in \mathcal{SK}$.
- $\sigma' \leftarrow \mathcal{SO}(n, L, pk_\pi, m)$. The Signing Oracle, on input a group size n, a set L of n public keys, the public key of the signer $pk_\pi \in L$, and a message m, returns a valid signature σ'.

If the scheme is proven in the random oracle model, a random oracle is simulated.

Nonslanderability. Nonslanderability ensures that no signer can generate a signature which is determined to be linked by the Linkability algorithm with another signature which is not generated by the signer. In other words, it prevents adversaries from framing honest users. Nonslanderability for LSAG schemes is defined in the following game between the Simulator $\mathcal{S}$ and the Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}, \mathcal{CO}, \mathcal{SO}$ and the random oracle:

1. $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters param.
2. $\mathcal{A}$ may query the oracles according to any adaptive strategy.
3. $\mathcal{A}$ gives $\mathcal{S}$ a group size n, a message m, a set of n public keys L, the public key of an insider $pk_\pi \in L$ such that pk_π has not been queried to $\mathcal{CO}$ or has not been included as the insider public key of any query to $\mathcal{SO}$. $\mathcal{S}$ uses the private key sk_π corresponding to pk_π to run the signature algorithm on input n, L, sk_π, m and to produce a signatures σ' given to $\mathcal{A}$.
4. $\mathcal{A}$ queries oracles with arbitrary interleaving. Except pk_π cannot be queried to $\mathcal{CO}$, or included as the insider public key of any query to $\mathcal{SO}$. In particular, $\mathcal{A}$ is allowed to query any public key which is not pk_π to $\mathcal{CO}$.
5. $\mathcal{A}$ delivers group size n^*, a set of n^* public keys L^*, a message m^* and a signature $\sigma^* \neq \sigma'$.

$\mathcal{A}$ wins the game if:

- Verification algorithm on input n^*, m^*, σ^* outputs "accept".
- All of the public keys in L^* are query outputs of $\mathcal{JO}$.
- pk_π has not been queried to $\mathcal{CO}$.
- The output of the Linking algorithm on input σ^*, σ is "linked".

The advantage probability of adversary $\mathcal{A}$

$$\mathbf{Adv}_{\mathcal{A}}^{NS}(\lambda) = Pr[\mathcal{A} \text{ wins the game}].$$

2.3 Proof of Nonslanderability

Theorem 1. *The LSAG is nonslanderable in the random oracle model, if the DL problem is hard.*

Proof. Let $\mathcal{A}$ be an adversary attacking the nonslanderability property of our LSAG scheme and $\mathcal{S}$ be a simulator of $\mathcal{A}$. It is assumed that $\mathcal{S}$ is an adversary against the hardness of the underlying discrete log problem. In this proof we show that breaking the nonslanderability property is as hard as breaking the DLG assumption. $\mathcal{A}$ submits queries to the given oracles $\mathcal{JO}, \mathcal{CO}, \mathcal{SO}$ with the restriction that it cannot query the corruption oracle $\mathcal{CO}$ on the public key pk_π.

$\mathcal{A}$ generates a list of public keys L which also includes the challenge public key pk_π and queries it's corrupt oracle on input pk_π together with the list L and a message m. The simulator $\mathcal{S}$ simulates this query as follows:

- If pi_π has been queried to $\mathcal{CO}$, $\mathcal{S}$ runs the signing algorithm on input L, sk_π, m, where the secret key sk_π is the output of the corruption oracle corresponding to pk_π and returns a signature σ to $\mathcal{A}$.
- IF π_π has not been queried to $\mathcal{CO}$, $\mathcal{S}$ runs the key generation algorithm on input 1^λ and obtains a list of public keys $L = \{y_1, \ldots, y_n\}$, where $y_i = g^{x_i}$ and x_i is the corresponding secret key for all $i \in [1, n]$. $\mathcal{S}$ computes $H_2(L) = h$ and $\tilde{y} = h^{x_\pi}$. It picks a random $u \in_R \mathbb{Z}_q$ and for all $i \in [1, n] \setminus \{\pi\}$ and computes $c_{\pi+1} = H_1(L, \tilde{y}, m, g^u, h^u)$ and $c_{i+1} = H_1(L, \tilde{y}, m, g^{s_i}, y_i^{c_i}, h^{s_i}\tilde{y}^{c_i})$. Finally $\mathcal{S}$ computes $s_\pi = u - x_\pi c_\pi$ and returns the signature $\sigma_L(m) = (c_1, s_1, \ldots, s_n, \tilde{y})$ to $\mathcal{A}$.
- Hash-function H_1, H_2 are programmed as random oracles. For all input which have been registered, $\mathcal{S}$ will return the corresponding output. Otherwise it will pick random values h_1, h_2. All queries to the random oracles are recorded in the corresponding lists by the simulator $\mathcal{S}$.

In the challenge step, $\mathcal{A}$ outputs a list of public keys L', a message m and a public key $pk'_\pi \in L'$ which has not been queried to the corrupt oracle $\mathcal{CO}$ nor to the signing oracle $\mathcal{SO}$. $\mathcal{S}$ computes a signature on this message m and outputs $\sigma'_L(m)$. $\mathcal{A}$ issues further queries except that it is not allowed to query the corrupt oracle on pk'_π.

$\mathcal{A}$ outputs a list of public keys L^* and a signature $\sigma^*_L(m^*)$ on a message m^* which passes the Verification algorithm and the Linking algorithm links the signatures $\sigma'_L(m), \sigma^*_L(m^*)$ for the lists L' and L^*. As showed above the simulator uses sk'_π to generate a signature σ'_L. If the Linking algorithm outputs "linked", then the public keys pk^*_π and pk'_π must be equal, i.e. the adversary $\mathcal{A}$ must know sk^*_π. Since $pk^*_\pi = pk'_\pi = g^{x^*_\pi}$ and the adversary is not allowed to query $\mathcal{CO}$ on pk'_π, $\mathcal{A}$ can not know sk^*_π, except it can break the discrete log assumption, which is a contradiction to the hardness of the underlying DL problem. □

References

1. Esgin, M.F., Zhao, R.K., Steinfeld, R., Liu, J.K., Liu, D.: Matrict: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In: ACM CCS 2019. pp. 567–584. ACM (2019)

2. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In: ACISP 2004. Lecture Notes in Computer Science, vol. 3108, pp. 325–335. Springer (2004)
3. Liu, J.K., Wong, D.S.: Linkable ring signatures: Security models and new schemes. In: Computational Science and Its Applications - ICCSA 2005, Part II. Lecture Notes in Computer Science, vol. 3481, pp. 614–623. Springer (2005)
4. Noether, S.: Ring signature confidential transactions for monero. IACR Cryptol. ePrint Arch. **2015**, 1098 (2015)
5. Sun, S., Au, M.H., Liu, J.K., Yuen, T.H.: Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: ESORICS 2017, Part II. Lecture Notes in Computer Science, vol. 10493, pp. 456–474. Springer (2017)
6. Torres, W.A.A., Kuchta, V., Steinfeld, R., Sakzad, A., Liu, J.K., Cheng, J.: Lattice ringct V2.0 with multiple input and multiple output wallets. In: ACISP 2019. Lecture Notes in Computer Science, vol. 11547, pp. 156–175. Springer (2019)
7. Torres, W.A.A., Steinfeld, R., Sakzad, A., Liu, J.K., Kuchta, V., Bhattacharjee, N., Au, M.H., Cheng, J.: Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1.0). In: ACISP 2018. Lecture Notes in Computer Science, vol. 10946, pp. 558–576. Springer (2018)
8. Tsang, P.P., Wei, V.K.: Short linkable ring signatures for e-voting, e-cash and attestation. In: ISPEC 2005. Lecture Notes in Computer Science, vol. 3439, pp. 48–60. Springer (2005)
9. Yuen, T.H., Sun, S., Liu, J.K., Au, M.H., Esgin, M.F., Zhang, Q., Gu, D.: Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security. In: FC 2020. Lecture Notes in Computer Science, vol. 12059, pp. 464–483. Springer (2020)