Triptych multisignature analysis

Cypher Stack

August 10, 2021

This document describes research and analysis into multisignature support for the Triptych zero-knowledge proving system and its associated transaction model.

All material in this document should be considered experimental and unsuitable for production without thorough independent review.

1 Introduction

Triptych [3] is a zero-knowledge proving system that can be used in a multidimensional linkable ring signature construction to build a confidential transaction protocol. Proofs constructed using Triptych scale in size logarithmically with the size of the input anonymity set, and can take advantage of efficient verification using batching and multiscalar multiplication.

A currently open question is the extent to which the protocol can support multisignature operations, where a group of players each holding a portion of a private signing key can jointly produce a valid transaction without mutual trust in each other or another third party. It is a requirement that the resulting proof be indistinguishable from one produced by a single player holding a private key; that is, the resulting proof must be verifiable using a standard public key.

A confidential protocol based on Triptych requires the use of a private key in two places: during generation of a one-of-many authorizing proof, and when constructing a linking tag used to prevent double-spend attempts. The one-of-many proof uses the private key in a linear fashion, which allows for straightforward approaches to multiparty computation. However, the linking tag is constructed using the inverse of the private key, posing a challenge to such operations. Fortunately, this inversion operation is similar to that used in other protocols, so we can apply earlier work toward a solution.

Let $\mathbb{G}$ be the prime-order subgroup of the `ed25519` elliptic curve group, and let $\mathbb{F}$ be its scalar field. Let G, H, and U be independent fixed generators of $\mathbb{G}$ known to all players. Let $\mathcal{H} : \{0,1\}^* \to \mathbb{F}$ be a cryptographic hash function. We often prefix input to $\mathcal{H}$ with an integer value, to act as an arbitrary domain separator; note that any equivalent domain separation method may be substituted, and that our notation is used for convenience. When referring

to sets of keys generated by players, we assume a fixed and publicly-known ordering, such as lexicographic ordering, is used.

2 Key generation

Keys are generated in a manner similar to that of MuSig [2]. We assume a set of ν players who wish to generate linear shares of a private and public key constructed in the following way. Throughout this section, we assume actions taken by a player $\alpha \in [0, \nu)$ are separately taken by all ν players in parallel.

1. Each player α chooses random $(a_\alpha, b_\alpha) \in \mathbb{F}^2$ and computes $B_\alpha \equiv b_\alpha G$. It generates a private key γ_α and corresponding public key Γ_α for the Paillier cryptosystem compatible with encryption of `ed25519` prime-order subgroup scalar elements. It sends the tuple $(a_\alpha, B_\alpha, \Gamma_\alpha)$ to all players.

2. Each player α computes the aggregate private view key

$$a \equiv \sum_{\alpha=0}^{\nu-1} \mathcal{H}(0, \{a_\beta G\}_{\beta=0}^{\nu-1}, \alpha) a_\alpha$$

   and corresponding aggregate public view key

$$A \equiv aG = \sum_{\alpha=0}^{\nu-1} \mathcal{H}(0, \{a_\beta G\}_{\beta=0}^{\nu-1}, \alpha) a_\alpha G.$$

3. Each player α computes the aggregate public spend key

$$B \equiv \sum_{\alpha=0}^{\nu-1} \mathcal{H}(1, \{B_\beta\}_{\beta=0}^{\nu-1}, \alpha) B_\alpha.$$

The aggregate public key pair (A, B) is used to receive transactions.

3 Transaction generation and scanning

A sender produces a transaction for the address (A, B) as usual; we describe the process here for the sake of notation clarity.

1. The sender chooses random $r \in \mathbb{F}$ and sets $R \equiv rG$.

2. The sender computes the output public key $P \equiv \mathcal{H}(2, rA)G + B$.

3. The sender chooses random $s_1 \in \mathbb{F}$ and, using a valid value $v \in \mathbb{F}$, computes a commitment $C_{\text{val}} \equiv s_1 G + vH$.

4. The sender computes other auxiliary transaction information as needed.

The sender includes R, P, C_{val} among other public transaction information, and submits the transaction to the network.

Any of the ν players holding shares of the address (A, B) scan transactions for outputs destined for the address as usual, using the known aggregate private view key a and aggregate public spend key B. Specifically, for a given output public key P with associated R, they test if $\mathcal{H}(2, aR)G + B = P$, and ignore if the equality does not hold. If it does, we note the associated private key:

$$\begin{aligned} p &\equiv \mathcal{H}(2, aR) + b \\ &= \mathcal{H}(2, aR) + \sum_{\alpha=0}^{\nu-1} \mathcal{H}(1, \{B_\beta\}_{\beta=0}^{\nu-1}, \alpha) b_\alpha \end{aligned}$$

Of course, each player α holds only its own share b_α of the aggregate private spend key.

4 Linking tag computation

To spend the output represented by the public output key P, the players must cooperatively compute the associated linking tag J. In the Triptych protocol, this tag is defined (continuing our rather bespoke notation from this documentation, rather than directly using the notation from [3]) as follows:

$$\begin{aligned} J &\equiv p^{-1} U \\ &= \left(\mathcal{H}(2, aR) + \sum_{\alpha=0}^{\nu-1} \mathcal{H}(1, \{B_\beta\}_{\beta=0}^{\nu-1}, \alpha) b_\alpha \right)^{-1} U \end{aligned}$$

The non-linearity of the linking tag with respect to the associated output private key makes such a cooperative computation more complex. However, we can apply a method of Gennaro and Goldfeder [1], which was originally devised for use in multiparty threshold ECDSA signature computation. The result will be that all players obtain J without revealing their private spend key shares to each other.

Because p consists of an additive prefix $\mathcal{H}(2, aR)$ that is not distributed between players, we define players' secret shares $\{p_\alpha\}_{\alpha=0}^{\nu-1}$ of p according to their index:

- $p_0 := \mathcal{H}(2, aR) + \mathcal{H}(1, \{B_\beta\}_{\beta=0}^{\nu-1}, 0) b_0$
- $p_{\alpha>0} := \mathcal{H}(1, \{B_\beta\}_{\beta=0}^{\nu-1}, \alpha) b_\alpha$

The result is that $\sum_{\alpha=0}^{\nu-1} p_\alpha = p$. The designation of indices does not affect protocol security.

A more comprehensive version of this method assumes the use of range proofs coupled with Paillier ciphertext; however, the authors hypothesize (but do not prove) that no significant information leakage occurs if these proofs are omitted.

We specifically require the use of Paillier public-key encryption, the keys for which have been introduced previously. Let Enc_α represent Paillier encryption using the public key Γ_α for player α, and Dec_α represent Paillier decryption using the private key γ_α for this player.

The process proceeds as follows.

1. Each player α chooses random $g_\alpha \in \mathbb{F}$ and sets $G_\alpha \equiv g_\alpha U$. It generates a commitment $C_{\mathrm{tag},\alpha}$ to G_α and sends the commitment to all players.
2. Each player α sets $c_\alpha \equiv \mathrm{Enc}_\alpha(p_\alpha)$. It sends c_α to all players.
3. For each other player β, player α chooses random $b_{\alpha\beta} \in \mathbb{F}$. It sends the value $\gamma_\alpha c_\beta + \mathrm{Enc}_\beta(-b_{\alpha\beta})$ to player β, who decrypts using γ_β and defines $a_{\beta\alpha}$ to be the resulting plaintext.
4. Each player α computes $\delta_\alpha := p_\alpha\gamma_\alpha + \sum_{\beta\neq\alpha}(a_{\alpha\beta} + b_{\alpha\beta})$. It computes a Schnorr proof of representation Π_α of G_α with respect to the generator U. It sends G_α, Π_α, and δ_α to all players.
5. Each player computes $\delta := \sum_{\beta=0}^{\nu-1}\delta_\beta$. It ensures that the commitment $C_{\mathrm{tag},\beta}$ and Schnorr proof Π_β are valid for each β, and aborts otherwise. It then computes the aggregate linking tag $J := \delta^{-1}\sum_{\beta=0}^{\nu-1} G_\beta$.

Note that in this protocol, we use additive notation on Paillier ciphertexts, and multiplicative notation between field elements in $\mathbb{F}$ and Paillier ciphertexts. Because of Paillier encryption homomorphism, these operations are well defined.

5 Authorizing proof

Triptych requires the use of a one-of-many authorizing proof. When embedded with a linking tag and message using the non-interactive Fiat-Shamir heuristic, this proof serves as a digital signature that the signer knows the private key corresponding to an input key and corresponding to the linking tag. The proof effectively proceeds in two stages. In the first stage, the signer hides the index corresponding to the known public key in the input set. In the second stage, the signer demonstrates knowledge of the private key corresponding to the public key at the hidden index, and further shows that the linking tag is constructed correctly. Throughout this section, we assume the input set contains N key tuples, where $N = n^m$ for some $n, m \in \mathbb{N}$.

The proof proceeds as follows.

1. Any designated player assembles an input set $\{(M_{i,0}, M_{i,1})\}_{i=0}^{N-1}$. A secret index $0 \leq \ell < N$ is chosen. Each $M_{i,0} \in \mathbb{G}$ is an existing output public key, and the designated output public key is set such that $M_{\ell,0} = P$.
2. Any designated player computes a commitment offset $C_{\mathrm{off}} \equiv s_2 G + vH$, where $s_2 \in \mathbb{F}$ is chosen according to the transaction protocol as usual. Each $M_{i,1}$ is the amount commitment corresponding to $M_{i,0}$, offset by C_{off}. Let $s \equiv s_1 - s_2$, such that $M_{\ell,1} = sG$.

3. Each player α selects random

$$r_A^{(\alpha)}, r_B^{(\alpha)}, r_C^{(\alpha)}, r_D^{(\alpha)}, \left\{a_{j,i}^{(\alpha)}\right\}_{i=1,j=0}^{n-1,m-1}, \left\{\rho_j^{(\alpha)}\right\}_{j=0}^{m-1} \in \mathbb{F}$$

according to the proof protocol. It sends the values

$$r_A^{(\alpha)}, r_B^{(\alpha)}, r_C^{(\alpha)}, r_D^{(\alpha)}, \left\{a_{j,i}^{(\alpha)}\right\}_{i=1,j=0}^{n-1,m-1}, \left\{\rho_j^{(\alpha)} G\right\}_{j=0}^{m-1}, \left\{\rho_j^{(\alpha)} J\right\}_{j=0}^{m-1}$$

to all players.

4. Each player computes the following values:

$$\begin{aligned}
r_A &\equiv \sum_{\alpha=0}^{\nu-1} \mathcal{H}(3, \{r_A^{(\beta)}\}_{\beta=0}^{\nu-1}, \alpha) r_A^{(\alpha)} \\
r_B &\equiv \sum_{\alpha=0}^{\nu-1} \mathcal{H}(4, \{r_B^{(\beta)}\}_{\beta=0}^{\nu-1}, \alpha) r_B^{(\alpha)} \\
r_C &\equiv \sum_{\alpha=0}^{\nu-1} \mathcal{H}(5, \{r_C^{(\beta)}\}_{\beta=0}^{\nu-1}, \alpha) r_C^{(\alpha)} \\
r_D &\equiv \sum_{\alpha=0}^{\nu-1} \mathcal{H}(6, \{r_D^{(\beta)}\}_{\beta=0}^{\nu-1}, \alpha) r_D^{(\alpha)} \\
\{a_{j,i}\}_{i=1,j=0}^{n-1,m-1} &\equiv \left\{\sum_{\alpha=0}^{\nu-1} \mathcal{H}(7, \{a_{j,i}^{(\beta)}\}_{\beta=0}^{\nu-1}, \alpha) a_{j,i}^{(\alpha)}\right\}_{i=1,j=0}^{n-1,m-1} \\
\{\rho_j G\}_{j=0}^{m-1} &\equiv \left\{\sum_{\alpha=0}^{\nu-1} \mathcal{H}(8, \{\rho_j^{(\beta)} G, \rho_j^{(\beta)} J\}_{\beta=0}^{\nu-1}, \alpha) \rho_j^{(\alpha)} G\right\}_{j=0}^{m-1} \\
\{\rho_j J\}_{j=0}^{m-1} &\equiv \left\{\sum_{\alpha=0}^{\nu-1} \mathcal{H}(8, \{\rho_j^{(\beta)} G, \rho_j^{(\beta)} J\}_{\beta=0}^{\nu-1}, \alpha) \rho_j^{(\alpha)} J\right\}_{j=0}^{m-1}
\end{aligned}$$

It uses the index ℓ to compute the set $\{\sigma_{j,i}\}_{i,j=0}^{n-1,m-1}$ according to the proof protocol, and further uses this collection of values to compute A, B, C, D as well.

5. Each player computes the hash value μ, the values $\{X_j, Y_j\}_{j=0}^{m-1}$, and the challenge ξ according to the proof protocol. It computes the set $\{f_{j,i}\}_{i=1,j=0}^{n-1,m-1}$ and the values z_A, z_C, K.

6. Each player α computes

$$z^{(\alpha)} \equiv \mathcal{H}(1, \{B_\beta\}_{\beta=0}^{\nu-1}, \alpha) b_\alpha \xi^m - \sum_{j=0}^{m-1} \mathcal{H}(8, \{\rho_j^{(\beta)} G, \rho_j^{(\beta)} J\}_{\beta=0}^{\nu-1}, \alpha) \rho_j^{(\alpha)} \xi^j$$

and sends this value to all players.

7. Each player computes

$$z \equiv (\mathcal{H}(2, aR) + \mu s)\xi^m + \sum_{\alpha=0}^{\nu-1} z^{(\alpha)}$$

and assembles the complete proof.

6 Observations

A possible modification is to replace the Schnorr proof of representation of the value G_α in the Paillier inversion protocol with a proof that the associated commitment $C_{\text{tag},\alpha}$ has a known discrete logarithm.

It may be necessary to require a specific commitment round for the authorizing proof pre-challenge values. Currently, these values are computed using MuSig-type hash aggregation in order to assert that the resulting aggregated values are uniformly distributed and cannot be controlled by malicious players. However, the aggregation uses a common coefficient set to compute values of the form $\{\rho_j G\}$ and $\{\rho_j J\}$.

It may be secure in practice to compress the Paillier inversion protocol, sending the initial commitment and encrypted key shares simultaneously.

7 Conclusions

The use of cooperative signing in the Triptych transaction protocol imposes significant communication complexity. Notably, the inversion construction of linking tags requires an expensive many-to-many share conversion that itself requires multiple communication rounds between all signers, and a Paillier-based approach to this inversion itself requires support for arbitrary RSA groups.

Because security of the authorizing proof relies in part on Fiat-Shamir transcripts, signers cannot complete this proof (namely, the challenge values) without first computing the corresponding linking tag, as the tag must be embedded in the transcript. Further, it is not possible to complete the entire pre-challenge portion of the authorizing proof prior to the linking tag computation, as this also requires knowledge of the complete tag.

If efficient multisignature operations are a priority for which this level of communication complexity is infeasible, it may be beneficial to investigate alternate transaction protocols that use linear linking tag constructions and/or simpler cooperative authorization proofs. The author is aware of (and collaborating on the development of) at least two newer trustless protocols that take this approach: Seraphis* and Lelantus 3†.

It remains for stakeholders and interested parties to assess the feasibility of this construction as it applies to protocols of interest, and determine if Cypher

*`https://gist.github.com/UkoeHB/24f6be6125684046c5c4d9f49351bccf`

†Disclaimer: The Firo project funds Cypher Stack research that includes development of this protocol.

Stack or other researchers and developers should continue to formalize, optimize, implement, or deploy it in software.

References

[1] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1179–1194, New York, NY, USA, 2018. Association for Computing Machinery.

[2] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019.

[3] Sarang Noether and Brandon Goodell. Triptych: Logarithmic-sized linkable ring signatures with applications. In Joaquin Garcia-Alfaro, Guillermo Navarro-Arribas, and Jordi Herrera-Joancomarti, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 337–354, Cham, 2020. Springer International Publishing.