MantaPay Trusted Setup Protocol Specification

v0.0.0

Francisco Hernández Iglesias, Todd Norton *

September 23, 2022

Abstract

We describe the protocol for the MantaPay trusted setup ceremony to generate prover and verifier keys for Groth16 ZK-SNARK proofs.

Contents

*ordered alphabetically.

1 Introduction

The MantaPay protocol (ref. to the specs) guarantees transaction privacy by using the Groth16 [2] Non-Interactive Zero-Knowledge Proving System (NIZK). In short, Groth16 is defined over a bilinear pairing of elliptic curves $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$ over a prime field $\mathbb{F}_p$. Let $\phi \in \mathbb{F}_p^\ell$ denote the set of *public inputs*, $w \in \mathbb{F}_p^{m-\ell}$ the set of *witnesses*, whose knowledge we want to prove, and let $\tau \in (\mathbb{F}_p^*)^4$ be a set of randomly generated numbers known as the *simulation trapdoor*. Groth16 consists of four parts:

- $(\sigma, \tau) \leftarrow \mathsf{Setup}$: Randomly generates τ, from which it computes σ, which consists of elliptic curve points in $\mathbb{G}_1$ and $\mathbb{G}_2$. σ is to be understood as the proving and verifying keys for Groth16.
- $\pi \leftarrow \mathsf{Prove}(\sigma, \phi, w)$: Computes a proof of knowledge of w, π, for a given setup σ and public input ϕ.
- $0, 1 \leftarrow \mathsf{Verify}(\sigma, \phi, \pi)$: Checks whether the proof π is valid against the setup σ and the public input ϕ.
- $\pi \leftarrow \mathsf{Sim}(\tau, \phi)$: Simulates a proof that will always be valid when verified against the setup σ corresponding to τ and the public input ϕ.

It is important to note that the Sim function is what makes the Groth16 protocol zero-knowledge: you can compute a valid proof π for any given setup σ and public input ϕ without knowledge of the witness w, provided that you have access to the simulation trapdoor τ. But Sim also makes Groth16 potentially insecure: if a malicious agent knew τ for a given σ, they could fabricate valid proofs for any statement regardless of its veracity.

The goal of the *trusted setup* is to provide a Groth16 setup σ in a secure way, i.e., in such a way that nobody has access to the trapdoor τ that was used to compute it.

2 Context

2.1 Circuit

Throughout this paper, by circuit we mean a *Rank-1 Constraint System (R1CS)*. It is defined as a system of equations over $\mathbb{F}_r$ of the form

$$\sum_{i=0}^{m} a_i u_{i,q} \cdot \sum_{i=0}^{m} a_i v_{i,q} = \sum_{i=0}^{m} a_i w_{i,q}, \qquad q = 1, \dots, n, \tag{1}$$

where $a_0 = 1$. This system of equations, in the context of zero-knowledge proofs, is to be understood as follows:

- The numbers $u_{i,q}, v_{i,q}, w_{i,q}$ are constants in $\mathbb{F}_r$ which represent the operations performed in an arithmetic circuit. Here constant means constant in the protocol, e.g. MantaPay will have a fixed set of $u_{i,q}, v_{i,q}, w_{i,q}$.
- The numbers $\phi = (a_1, \dots, a_\ell)$ are the public inputs. In MantaPay, these correspond to the TransferPost, excluding the proof.
- The numbers $w = (a_{\ell+1}, \dots, a_m)$ are the witnesses. In MantaPay, these correspond to the elements of the Transfer which are not part of the TransferPost.
- An R1CS defines the following binary relation

$$R = \Big\{ (\phi, w) \Big| \ \phi = (a_1, \dots, a_\ell),\ w = (a_{\ell+1}, \dots, a_m),\ (1) \text{ is satisfied} \Big\} \subset \mathbb{F}_r^\ell \times \mathbb{F}_r^{m-\ell} \tag{2}$$

- The statements that can be proved in this terminology are of the form: for a given circuit (1) and public input ϕ, there exists[1] a witness w such that $(\phi, w) \in R$.

2.2 Quadratic Arithmetic Programs

Quadratic Arithmetic Programs (QAPs) give an alternative way to describe a circuit, equivalent to R1CS. A QAP is a system of polynomial equations of the form

$$\sum_{i=0}^{m} a_i u_i(X) \cdot \sum_{i=0}^{m} a_i v_i(X) \equiv \sum_{i=0}^{m} a_i w_i(X) \mod t(X), \tag{3}$$

where

[1] and I know

- $u_i(X), v_i(X), w_i(X) \in \mathbb{F}_r[X]$ are degree $n-1$ polynomials, and $t(X) \in \mathbb{F}_r[X]$ is a degree n polynomial, all of which are fixed for the protocol.
- The numbers $\phi = (a_1, \ldots, a_\ell)$ are the public inputs.
- The numbers $w = (a_{\ell+1}, \ldots, a_m)$ are the witnesses.
- A QAP defines the following binary relation

$$R = \left\{ (\phi, w) \middle| \ \phi = (a_1, \ldots, a_\ell),\ w = (a_{\ell+1}, \ldots, a_m),\ (3) \text{ is satisfied} \right\} \subset \mathbb{F}_r^\ell \times \mathbb{F}_r^{m-\ell} \tag{4}$$

- The statements that can be proved in this terminology are of the form: for a given circuit (3) and public input ϕ, there exists a witness w such that $(\phi, w) \in R$.

We derive the QAP description of a circuit from its R1CS description as follows:

1. Choose k as the minimal integer such that $2^k \geq n$. Fix $t(X) = X^{2^k} - 1$.
2. We derive the polynomial $u_i(X)$ from the R1CS vector $(u_{i,q})_{q=1}^n$ via a Lagrange basis $\{L_q(x)\}_{q \in Q}$ for the domain I. That is,
$$u_i(X) = \sum_{q=1}^{n} u_{i,q} L_q(x) \tag{5}$$
3. The reader may readily check that (1) is equivalent to (3) with these definitions.

2.3 The Groth16 Setup function

Let us now recall how the Groth16 Setup function works in the MantaPay protocol. We start with

- The pairing curve BN254 [?], which consists of a triple of elliptic curves $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We fix generators g and h of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Note that $\mathrm{ord}(g) = \mathrm{ord}(h) = r$ is prime, and that $e(g, h)$ generates a subgroup of order r of $\mathbb{G}_T \cong F_{p^{12}}^*$.
- The MantaPay circuit, encoded as a QAP $\{u_i(X), v_i(X), w_i(X), t(X)\}$.

All public parameters derive from a simulation trapdoor $\tau = (\alpha, \beta, \delta, x) \leftarrow \mathbb{F}_r^*$ (the famous "toxic waste"). The Groth16 public parameters themselves are elements of $\mathbb{G}_1$ and $\mathbb{G}_2$, specifically

$$\sigma_1 = \left[\left(\alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \left\{\beta u_i(x) + \alpha v_i(x) + w_i(x)\right\}_{i=0}^{\ell}, \left\{ \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} \right\}_{i=\ell+1}^{m}, \left\{ \frac{x^i t(x)}{\delta} \right\}_{i=0}^{n-2} \right) \right]_1$$

$$\sigma_2 = \left[\left(\beta, \delta, \{x^i\}_{i=0}^{n-1} \right) \right]_2$$

where $[y]_1 = y \cdot g$ and $[y]_2 = y \cdot h$ for all $y \in \mathbb{F}_r$.

The output of Setup is $\sigma = (\sigma_1, \sigma_2)$. These are the public parameters from which Groth16 proofs are formed. The trapdoor τ is *not* public; indeed, a malicious prover with knowledge of τ could construct fraudulent proofs. The goal of the trusted setup is to generate σ without revealing τ.

2.4 Multi-Party Computation

A decentralized way to generate σ without revealing τ is to compute σ in such a way that τ becomes a shared secret split among a diverse set of participants. This may be achieved via *secure multi-party computation* (MPC). The MPC we employ is a protocol for computing σ incrementally from private inputs τ_i belonging to participants P_i in the computation.

The key security property of this MPC is that its security is ensured by having at least one honest participant. An honest participant is one who keeps their private input τ_i from all other participants, ideally by permanently clearing it from their system's memory after participation. Put differently, this *1-out-of-N* honest participants guarantee states that to determine the toxic waste τ requires the collusion of *all* participants in the MPC.

By soliciting contributions to the MPC from a diverse set of participants, we increase the difficulty of such collusion. Note that any individual with a stake in the security of MantaPay can guarantee this personally, simply by participating honestly in the Setup MPC.

(? Maybe mention verifiability of the MPC transcript?)

2.5 Phase Structure

The full Setup MPC splits usefully into two *phases*. In *Phase 1* we generate a modified KZG setup (todo cite KZG) which is *universal* in the sense that these parameters may be used by any ZK circuit of small enough size. In *Phase 2* we derive σ from the output of Phase 1. The parameters generated in Phase 2 are *circuit-specific*: they depend on the QAP description of the circuit (3) and must be computed separately for each ZK circuit. This two-phase splitting of the MPC is formalized in [1].

2.5.1 Phase 1

The first class consists of those which only depend on the elliptic curves and not on the circuit, i.e., on the QAP. These parameters, namely (TODO: Get the number of powers of $[x^i]_1$ right)

$$KZG = (\{[x^i]_1\}_{i=0}^{2n-2}, \{[\alpha x^i]_1\}_{i=0}^{n-1}, \{[\beta x^i]_1\}_{i=0}^{n-1}, [\beta]_2, \{[x^i]_2\}_{i=0}^{n-1}) \tag{6}$$

are known as the *Kate–Zaverucha–Goldberg (KZG)* commitments. Since these parameters don't depend on the specifics of the circuit, we take them from the Perpetual Powers of Tau (PPoT) project [**?**], a multi-party computation similar to our trusted setup described below, with a 1-out-of-N security assumption. To ensure its soundness, we have personally verified each contribution to PPoT.

2.5.2 Phase 2

The rest of the Groth16 parameters do depend on the circuit, so we have the corresponding values for the MantaPay circuit. The rest of the paper is devoted to explaining in detail the multi-party computation known as the trusted setup ceremony which we perform to make sure the remaining Groth16 parameters are computed safely.

- Refer to above definition of σ
- Explain ininitialization: at least mention that inputs are (6) plus QAP coefficients.
- Mention that at this point we have prover keys but with $\delta = 1$, and the point is to modify δ.
- Outline how σ is derived from these inputs. This part is MPC, reference protocol description below.

Also need to define what is a phase 2 Contribution, what is a Proof. (Maybe this is in Protocol Design?)

3 Requirements

3.1 Goals

- Anonymity: Pre-registration only requires a Twitter handle and an email address, none of which needs to be linked to your identity.
- Server coordination: The ceremony will be coordinated by a server, which will:
  - Manage the contribution queue for the participants, supporting priority tiers.
  - Ensure that only pre-registered participants with valid signatures are allowed to take part in the ceremony.
  - Ensure that no more than one participant is contributing at a given time.
  - Ensure that no participant contributes more than once.
  - Inform the participants about their queue/contribution status.
  - Check the validity proof of the proposed contributions before adding them to the ceremony.
  - After a successful contribution, return its hash to the participant.
- Updatable: More contributions can be added to the ceremony at later stages if desired.
- Verifiable: The ceremony and all its contributions can be verified by independent auditors. We ensure that is the case by:
  - Publishing the transcript of the ceremony, including the validity proof and hash of each contribution.
  - Asking participants to publish their hashes in independent locations, e.g. Twitter.
  - Distributing an open-source verification library.

3.2 Non-Goals

- Permisionless: We require participants to pre-register to contribute to the ceremony. However, registration is open to anyone with a Twitter account and an email address.
- Support for independently computed contributions: The ceremony only supports those contributions done through the server with our client. However, the server code is open-source and the transcript of the ceremony is publicly available for audits.

4 Design

4.1 Ceremony Protocol

Has 2 parties: Contributor, Coordinator. They exchange messages according to MessagingProtocol.

Protocol has Initialization phase and Contribution phase.

Input to Initialization phase:

- KZG Phase 1 param.s (ref. def.)
- circuit description(s) (ref. def.).

Output of Initialization phase:

- Groth16 ProverKey (ref. def.) PK_0
- Initial Challenge $Chall_0$

Contribution phase consists of repeated Rounds, each of which looks like:

1. Coordinator send (PK, chall)(n) to Contributor(n).
2. Contributor calls Contribute:
   (a) Sample Randomness
   (b) Compute contribution (ref. def.)
   (c) Generate proof (order?)
3. Contributor sends (PK, proof)(n+1) to Coordinator (signed message verification occurs)
4. Coordinator calls Verify:
   (a) Compute chall(n+1) from PK, chall(n)? (anything else?)
   (b) Verify(proof(n+1), chall(n+1), PK(n), PK(n+1))
5. Coordinator archives PK, chall, proof.

4.2 Messaging Protocol

TODO

4.3 Server State Machine

Responsibilities:

- Parameter Initialization (see above)
- Contribution verification
- Signature verification
- Registry maintenance
- Queue management
- Contribution archive

State:

- Registry: Registry

- MpcState: (ProverKey x Challenge x Proof) x no. circuits (TODO: How explicitly should be mention the multiple parallel circuits?)
- Transcript: (ditto) x no. rounds

Methods:

- Initialize
- Enqueue
- ParseMessage
- VerifyContribution
- SendMessage

Operation:

1. Initialize
2. Concurrent Loop: Coordinate Contributions (TODO: Flow chart)
   (a) ParseMessage (contribute).
   (b) Consult Registry, SendMessage (either QueuePosition or MpcState)
   (c) Await ContributionResponse, ParseMessage to yield Contribution
   (d) Verify Contribution
   (e) Update MpcState, update Registry
   (f) Add Contribution to Transcript
3. Concurrent Loop: Queue Management
   (a) ParseMessage (JoinQueue), yield ParticipantId
   (b) Consult Registry
   (c) Enqueue (if Participant has not contributed)

4.4 Client State Machine

Responsibilities:

- Ed25519 keypair generation
- Message Signing
- Contribution Computation

State:

- Keypair: SignatureKeypair
- Randomness: PRNG

Methods:

- GenerateKeypair
- RequestState
- Contribute

Operation:

1. GenerateKeypair, initialize own state
2. RequestState: send signed message asking for the MpcState
3. Await StateResponse
4. Contribute, send signed ContributeRequest
5. Output Attestation

(Mention nonce management? Or is that an internal detail to signatures?)

5 References

References

[1] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. Cryptology ePrint Archive, Paper 2017/1050, 2017. https://eprint.iacr.org/2017/1050.

[2] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.