Zero-Balance Argument

1 Preliminaries

Basic notation. For two integers $n < m$, we write $[n, m]$ to denote the set $\{n, n+1, \ldots, m\}$. When $n = 1$, we simply write $[m]$ to denote the set $\{1, \ldots, m\}$. For any finite set S, we use $x \leftarrow_{\mathsf{R}} S$ to denote the process of sampling an element $x \in S$ uniformly at random. Unless specified otherwise, we use λ to denote the security parameter. We say that an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. We say that a function $f : \mathbb{N} \to \mathbb{N}$ is negligible if $f = o(1/n^c)$ for any positive integer $c \in \mathbb{N}$. Throughout the exposition, we use $\mathsf{poly}(\cdot)$ and $\mathsf{negl}(\cdot)$ to denote any polynomial and negligible functions respectively.

1.1 Discrete Log Relation Assumption

The discrete log relation assumption states that given a number of random group elements in $\mathbb{G}$, no efficient adversary can find a non-trivial relation on these elements.

Definition 1.1 (Discrete Log Relation)**.** Let $\mathbb{G} = \mathbb{G}(\lambda)$ be a group of prime order p. Then the *discrete log relation* assumption on $\mathbb{G}$ states that for any efficient adversary $\mathcal{A}$ and $n \geq 2$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr\left[\mathcal{A}(G_1, \ldots, G_n) \to a_1, \ldots, a_n \in \mathbb{Z}_p : \exists\ a_i \neq 0 \wedge \sum_{i \in [n]} a_i \cdot G = 0\right] = \mathsf{negl}(\lambda),$$

where $G_1, \ldots, G_n \leftarrow_{\mathsf{R}} \mathbb{G}$.

1.2 Rewinding Lemma

To prove security, we make use of the rewinding lemma. For the purpose of this document, we do not require the rewinding lemma in its full generality and therefore, we rely on the following simple variant from Boneh and Shoup [1].

Lemma 1.2 (Rewinding Lemma)**.** *Let S and T be finite, non-empty sets, and let $f : S \times T \to \{0, 1\}$ be a function. Let X, Y, and Y' be mutually independent random variables, where X takes values in the set S, and Y, Y' are each uniformly distributed over T. Let $\varepsilon = \Pr\left[f(X, Y) = 1\right]$ and $N = |T|$. Then*

$$\Pr\left[f(X, Y) = 1 \wedge f(X, Y') = 1 \wedge Y \neq Y' big\right] \geq \varepsilon^2 - \varepsilon/N.$$

2 Zero-Knowledge Argument Definitions

In full generality, zero-knowledge argument systems can be defined with respect to any class of decidable languages. However, to simplify the presentation, we define argument systems with respect to CRS-dependent languages. Specifically, let $\mathcal{R} \subset \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ be an efficiently decidable ternary relation. Then a CRS-dependent language for a string $\rho \in \{0,1\}^*$ is defined as

$$\mathcal{L}_\rho = \{u \mid \exists\ w : (\rho, u, w) \in \mathcal{R}\}.$$

We generally refer to ρ as the common reference string, u as the instance of the langauge, and w as the witness for u.

For a class of CRS-dependent languages, an argument system consists of the following algorithms.

Definition 2.1 (Argument System)**.** A non-interactive argument system Π_{AS} for a CRS-dependent relation $\mathcal{R}$ consists of a tuple of efficient algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \rho$: On input the security parameter λ, the setup algorithm returns a common reference string ρ.
- $\mathcal{P}(\sigma, u, w)$: The prover $\mathcal{P}$ is an interactive algorithm that takes in as input a common reference string σ, instance u, and witness w. It interacts with the verifier $\mathcal{V}$ according to the specification of the protocol.
- $\mathcal{V}(\sigma, u)$: The verifier $\mathcal{V}$ is an interactive algorithm that takes in as input a common reference string ρ and an instance x. It interacts with the prover $\mathcal{P}$ in the protocol and in the end, it either accepts (returns 1) or rejects (returns 0) the instance x.

We use $\langle \mathcal{P}(\rho, u, w), \mathcal{V}(\rho, u) \rangle = 1$ to denote the event that the verifier $\mathcal{V}$ accepts the instance of the protocol. We use $\langle \mathcal{P}(\rho, u, w), \mathcal{V}(\rho, u) \rangle \to \mathsf{tr}$ to denote the communication transacript between the prover $\mathcal{P}$ and verifier $\mathcal{V}$ during a specific execution of the protocol.

An argument system must satisfy a correctness and two security properties. The correctness property of an argument system is generally referred to as *completeness*. It states that if the prover $\mathcal{P}$ takes in as input a valid instance-witness tuple $(\rho, u, w) \in \mathcal{R}$ and follows the protocol specification, then it must be able to convince the verifier to accept.

Definition 2.2 (Completeness)**.** Let Π_{AS} be a proof system for a relation $\mathcal{R}$. Then we say that Π_{AS} satisfies perfect completeness if for any $(u, w) \in \mathcal{R}$, we have

$$\Pr\left[\langle \mathcal{P}(\rho, u, w), \mathcal{V}(\rho, u) \rangle = 1\right] = 1,$$

where $\rho \leftarrow \mathsf{Setup}(1^\lambda)$.

The first security property that an argument system must satisfy is *soundness*, which can be defined in a number of ways. In this work, we work with *computational witness-extended emulation* as presented in Bulletproofs [2].

Definition 2.3 (Soundness [3, 4, 2])**.** Let Π_{AS} be a proof system for a relation $\mathcal{R}$. Then we say that Π_{AS} satisfies *witness-extended emulation* soundness if for all deterministic polynomial time $\mathcal{P}^*$,

there exists an efficient emulator $\mathcal{E}$ such that for all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\left| \Pr\left[\mathcal{A}_2(\mathsf{tr}) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathsf{Setup}(1^\lambda), (u, \mathsf{st}) \leftarrow \mathcal{A}_1(\rho), \\ \mathsf{tr} \leftarrow \langle \mathcal{P}^*(\rho, u, \mathsf{st}), \mathcal{V}(\rho, u) \rangle \end{array}\right] - \Pr\left[\mathcal{A}_2(\mathsf{tr}) = 1 \wedge (\mathsf{tr} \text{ accepting } \Rightarrow (\rho, u, w) \in \mathcal{R}) \;\middle|\; \begin{array}{c} \rho \leftarrow \mathsf{Setup}(1^\lambda), \\ (u, \mathsf{st}) \leftarrow \mathcal{A}_1(\rho), \\ (\mathsf{tr}, w) \leftarrow \mathcal{E}^{\mathcal{O}}(\rho, u) \end{array}\right] \right| = \mathsf{negl}(\lambda),$$

where the oracle is defined as $\mathcal{O} = \langle \mathcal{P}^*(\rho, u, \mathsf{st}), \mathcal{V}(\rho, u) \rangle$. The oracle $\mathcal{O}$ allows the emulator $\mathcal{E}$ to rewind the protocol to a specific point and resume the protocol after reprogramming the verifier with fresh randomness.

Traditionally, the soundness condition for an argument system of knowledge requires that there exists an extractor that can use its rewinding capability to extract a valid witness from any accepting transcript of the protocol that is produced by a dishonest prover $\mathcal{P}^*$. The witness-extended emulation strengthens this traditional definition by requiring that the extractor (emulator) not only successfully extracts a valid witness, but also produces (emulates) a valid transcript of the protocol for which the verifier accepts. The value st in the definition above can be viewed as the internal state of $\mathcal{P}^*$, which can also be its randomness.

The second security property that we require from an argument system is the zero-knowledge property. All argument systems that we rely on in the `ZK-Token` program are public coin protocols that we ultimately convert into a non-interactive protocol. Therefore, we rely on the standard zero-knowledge property against honest verifiers.

Definition 2.4 (Zero-Knowledge)**.** Let Π_{AS} be a proof system for a relation $\mathcal{R}$. Then we say that Π_{AS} satisfies *honest verifier* zero-knowledge if there exists an efficient simulator $\mathcal{S}$ such that for all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have

$$\Pr\left[(\rho, u, w) \in \mathcal{R} \wedge \mathcal{A}_1(\mathsf{tr}) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathsf{Setup}(1^\lambda), (u, w, \tau) \leftarrow \mathcal{A}_2(\rho), \\ \mathsf{tr} \leftarrow \langle \mathcal{P}(\rho, u, w), \mathcal{V}(\rho, u; \tau) \rangle \end{array}\right]$$

$$= \Pr\left[(\rho, u, w) \in \mathcal{R} \wedge \mathcal{A}_1(\mathsf{tr}) = 1 \;\middle|\; \begin{array}{c} \rho \leftarrow \mathsf{Setup}(1^\lambda), \\ (u, w, \tau) \leftarrow \mathcal{A}_2(\rho), \\ \mathsf{tr} \leftarrow \mathcal{S}(u, \tau) \end{array}\right],$$

where ρ is the public coin randomness used by the verifier.

3 Argument System Description

The zero-balance protocol allows a prover to convince the verifier that a twisted ElGamal ciphertext encrypts the value zero under a specified public key. Formally, the zero-balance protocol captures the following language:

$$\mathcal{L}^{\mathsf{zero\text{-}balance}}_{G,H} = \left\{ u = (P, C, D) \in \mathbb{G}^3, w = s \in \mathbb{Z}_p \;\middle|\; s \cdot P = H \wedge s \cdot D = C \right\}.$$

The language is defined with respect to two fixed generators G, H that defines the twisted ElGamal encryption scheme. The group element P corresponds to a public key in the encryption scheme

and the field element s corresponds to its secret key. The elements C, D correspond to a Pedersen commitment and decryption handle that make up a single ciphertext. If the ciphertext $\mathsf{ct} = (C, D)$ is a proper encryption of zero, then its decryption must produce $\mathsf{Decrypt}(s, \mathsf{ct}) = C - s \cdot D = 0 \cdot G = 0$ and hence $s \cdot D = C$. The zero-balance argument system for the language $\mathcal{L}_{G,H}$ is specified as follows:

Prover(u, w)		**Verifier**(u)
$y \leftarrow_{\mathsf{R}} \mathbb{Z}_p$		
$Y_P \leftarrow y \cdot P$ $Y_D \leftarrow y \cdot D$		
	$\xrightarrow{Y_P,\ Y_D}$	
		$c \leftarrow_{\mathsf{R}} \mathbb{Z}_p$
	$\xleftarrow{c}$	
$z \leftarrow c \cdot s + y$		
	$\xrightarrow{z}$	
		$z \cdot P \stackrel{?}{=} c \cdot H + Y_P$ $z \cdot D \stackrel{?}{=} c \cdot C + Y_D$

The protocol follows a standard sigma protocol structure where the prover first samples a random field element $y \leftarrow_{\mathsf{R}} \mathbb{Z}_p$. It then commits to this element by sending $Y_P = y \cdot P$ and $Y_D = y \cdot D$ to the verifier. Upon receiving a random challenge c, it provides the verifier with the masked secret key $z = c \cdot s + y$. Finally, the verifier tests the two relations $s \cdot P = H$ and $s \cdot D = C$ using the masked secret key z and the committed values Y_P and Y_D.

The zero-balance argument system above satisfies all the correctness and security properties that are specified in Section 2. We formally state these properties in the following theorems.

Theorem 3.1 (Completeness)**.** *The zero-balance argument satisfies completeness 2.2.*

Theorem 3.2 (Soundness)**.** *Suppose that $\mathbb{G}$ is a prime order group for which the discrete log relation assumption (Definition 1.1) holds. Then the zero-balance argument satisfies witness-extended emulation soundness 2.3.*

Theorem 3.3 (Zero-Knowledge)**.** *The zero-balance argument satisfies perfect honest verifier zero-knowledge 2.4.*

We provide the formal proofs for these theorems in Section 4.

4 Proof of Security

4.1 Proof of Theorem 3.1

To prove completeness, let us fix any valid instance and witness for $\mathcal{L}_{G,H}^{\mathsf{zero-balance}}$: $P, C, D \in \mathbb{G}$ and $s \in \mathbb{Z}_p$ such that $s \cdot P = H$ and $s \cdot D = C$. It suffices to show that after an honest execution of the

protocol by the prover, the verifier always returns 1 at the end of the protocol. Let y and c be any elements in $\mathbb{Z}_p$ and let $Y_P = y \cdot P$, $Y_D = y \cdot D$, and $z = c \cdot s + y$ in an execution of the protocol. Then we have

$$z \cdot P = (c \cdot s + y)P = c \cdot (s \cdot P) + y \cdot P = c \cdot H + Y_P,$$
$$z \cdot D = (c \cdot s + y)D = c \cdot (s \cdot D) + y \cdot D = c \cdot C + Y_D.$$

As both of the algebraic relations that the verifier checks at the end of the protocol hold, the proof is always accepted. Completeness follows.

4.2 Proof of Theorem 3.2

To prove soundness, we construct an emulator $\mathcal{E}$ that has oracle access to any malicious prover $\mathcal{P}^*$ and extracts a valid witness by rewinding $\mathcal{P}^*$ and simulating two execution of the zero-balance protocol with an honest verifier $\mathcal{V}$.

Let (P, C, D) be an instance of the language $\mathcal{L}^{\mathsf{zero-balance}}_{G,H}$. We construct an emulator $\mathcal{E}$ that uses $\mathcal{P}^*$ to extract a valid witness as follows:

- It runs $\mathcal{P}^*$ to receive the first message (Y_P, Y_D).
- It maintains a list L of accepting transcripts, initially empty.
- It then loops, each time rewinding $\mathcal{P}^*$ to the point before the verifier sends the challenge c:
  1. It provides fresh randomness to the verifier $\mathcal{V}$ to get a new challenge c_i.
  2. It completes the protocol to get the prover's responses z_i.
  3. It checks if the transcript $\mathsf{tr}_i = (Y_P, Y_D, c_i, z_i)$ is accepting by verifying if $z_i \cdot P = c_i \cdot H + Y_P$ and $z_i \cdot D = c_i \cdot C + Y_D$.
  4. If tr_i is accepting, it stores (c_i, z_i) in its list L.
- It continues this process utnil it has stored two accepting transcripts, (c_j, z_j) and (c_k, z_k) such that $c_j \neq c_k$.
- Once it has such a pair, it computes the witness $s \leftarrow (z_j - z_k)/(c_j - c_k)$. It then returns (tr_j, s).

To complete the proof, we first bound the probability that $\mathcal{E}$ does not abort at the end of the two executions of $\langle \mathcal{P}^*(\rho, u, \mathsf{st}), \mathcal{V}(\rho, u) \rangle$. Then, we show that if $\mathcal{E}$ does not abort, then the extracted witness $s = (z - z')/(c - c')$ is a valid witness.

Abort probability. The emulator $\mathcal{E}$ aborts only when $c = c'$, which is dependent on the probability that $\mathcal{P}^*$ successfully convinces $\mathcal{V}$ at the end of the protocol. Let ε_{P^*} be the probability that $\mathcal{P}^*$ successfully convinces $\mathcal{V}$ in $\langle \mathcal{P}^*(\rho, u, \mathsf{st}), \mathcal{V}(\rho, u) \rangle$. We bound the probability that $c = c'$ with ε_{P^*} using the rewinding lemma 1.2. Specifically, let us define the following random variables:

- Let X be the elements (Y_P, Y_D) in the transcript of an execution of $\langle \mathcal{P}^*(\rho, u, \mathsf{st}), \mathcal{V}(\rho, u) \rangle$.
- Let Y and Y' be the values c and c' respectively in the two executions of $\langle \mathcal{P}^*(\rho, u, \mathsf{st}), \mathcal{V}(\rho, u) \rangle$.
- Let Z and Z' be the values z and z' respectively in the two executions of $\langle \mathcal{P}^*(\rho, u, \mathsf{st}), \mathcal{V}(\rho, u) \rangle$.

- Let $f(\mathsf{tr}) \to \{0,1\}$ be the protocol verification function that returns 1 if tr is an accepting transcript and 0 otherwise.

Then, the rewinding lemma states that

$$\Pr\left[f(X,Y,Z) = 1 \wedge f(X,Y',Z') = 1 \wedge Y \neq Y'\right] \geq \varepsilon_{P^*}^2 - \varepsilon_{P^*}/p.$$

By assumption, we have $1/p = \mathsf{negl}(\lambda)$. Therefore, if ε_{P^*} is non-negligible, then the probability that ε succeeds in finding two distinct accepting transcripts (i.e., does not abort) is non-negligible.

Witness validity. Now assume that the two executions of $\langle \mathcal{P}(\rho,u,w), \mathcal{V}(\rho,u)\rangle$ returns two accepting transcripts $\mathsf{tr} = (Y_P, Y_D, c, z)$, $\mathsf{tr}' = (Y_P, Y_D, c', z')$, and that $\mathcal{E}$ does not abort and returns $s = (z - z')/(c - c')$. Since tr and tr' are accepting transcripts, we have

$$z \cdot P = c \cdot H + Y_P,$$

$$z' \cdot P = c' \cdot H + Y_P.$$

This means that $(z - z') \cdot P = (c - c') \cdot H$ and hence, $s \cdot P = H$ where $s = (z - z')/(c - c')$. Similarly, we have

$$z \cdot D = c \cdot C + Y_D,$$

$$z' \cdot D = c' \cdot C + Y_D.$$

Therefore, the relation $(z - z') \cdot D = (c - c') \cdot C$ holds, which means that $s \cdot D = C$.

We have shown that if $\mathcal{P}^*$ successfully convinces the verifier $\mathcal{V}$ for an instance $x = (P, C, D)$ with non-negligible probability, then the emulator $\mathcal{E}$ successfully extracts a valid witness s. This completes the proof of soundness.

4.3 Proof of Theorem 3.3

Fix any elements $P, C, D \in \mathbb{G}$ and $s \in \mathbb{Z}_p$ such that $s \cdot P = H$ and $s \cdot D = C$. Let $\mathsf{tr}^* = (Y_P^*, Y_D^*, c^*, z^*)$ be any accepting transcript. By the specification of the protocol, the probability that an honest execution of the protocol by the prover and the verifier results in the transcript tr^* is as follows:

$$\Pr\left[\langle \mathcal{P}(\rho,u,w), \mathcal{V}(\rho,u)\rangle \to \mathsf{tr} \wedge \mathsf{tr} = \mathsf{tr}^*\right] = 1/p^2.$$

To prove zero-knowledge, we define a simulator $\mathcal{S}$ that produces such distribution without knowledge of a valid witness s.

$\mathcal{S}(P, C, D)$:

1. Sample $c, z \leftarrow_{\mathsf{R}} \mathbb{Z}_p$.
2. Set $Y_P = z \cdot P - c \cdot H$.
3. Set $Y_D = z \cdot D - c \cdot C$.
4. Return $\mathsf{tr} = (Y_P, Y_D, c, z)$.

The simulator $\mathcal{S}$ returns a transcript that is uniformly random under the condition that $z \cdot P = Y_P + c \cdot D$ and $z \cdot D = Y_D + c \cdot C$. As the variables Y_P, Y_D are completely determined by c, z, we have

$$\Pr\left[\mathcal{S}(P,C,D) \to \mathsf{tr} \wedge \mathsf{tr} = \mathsf{tr}^*\right] = 1/p^2,$$

for any fixed transcript tr^*. Zero-knowledge follows.

References

[1] Boneh, D., and Shoup, V. A graduate course in applied cryptography, 2020.

[2] Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., and Maxwell, G. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)* (2018), IEEE, pp. 315–334.

[3] Groth, J., and Ishai, Y. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2008), Springer, pp. 379–396.

[4] Lindell, Y. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology 16*, 3 (2003).