Extending Succinct Zero Knowledge Proofs for Set Membership to Ring Signatures

Jialin Li

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-152
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-152.html

May 20, 2022

Abstract

Extending Succinct Zero Knowledge Proofs for Set Membership to Ring Signatures

by

Jialin Li

in

University of California, Berkeley

,

Ring signatures are digital signatures that confirm the signer being a member of a public group without revealing the identity of the signer. There are wide applications of ring signatures, such as in the blockchain space. For example, anonymous cryptocurrency Monero employs RingCT, which is based on Confidential Transactions and ring signatures. The bottleneck for ring signatures is the size of the signatures as most current schemes have size of signatures proportional or logarithmic to the number of parties in the ring, which are highly inefficient. We hope to extend a state-of-art disjunctive zero-knowledge proof to ring signatures.

Disjunctive zero knowledge proofs, where the prover demonstrates knowledge of solution to a subset of problems, were first studied by Cramer et al.[8] Since then, there have been numerous optimizations towards more efficient communication and computation. In order to reach amortized computation that does not grow with the total number of statements, [6] uses RSA set accumulators to combine with commitment schemes in a $\Sigma-$Protocol. There is a natural connection between ring signatures and $\Sigma-$protocols as any $\Sigma-$protocol of zero knowledge proofs can be converted to a ring signatures by embedding the message to be signed in the hash function. Therefore, the construction of [6] give a ring signature scheme, which can be potentially applied to the Monero protocol after modifications.

Contents

Acknowledgments

I thank Sanjam Garg for his advice on this report. I also thank Alexander Bienstock, Aarushi Goel, and Abhishek Jain for discussions on topics related to this report.

Chapter 1

Commit-and-Prove Zero knowledge

1.1 Zero Knowledge Proof of Partial Knowledge

Zero Knowledge Proofs Zero Knowledge proofs, first proposed by Goldwasser, Micali and Rickoff in 1985 [12], are used to prove the validity of a statement without leaking any additional information. Essentially, any information the verifier learns by interacting with the prover can be learned by the verifier on its own.

- Completeness If the statement is true ($x \in L$), an honest verifier will be convinced by the prover.
- Soundness If the statement is false ($x \notin L$), no prover should be able to convince the verifier.
- Zero knowledge For any verifier strategy V^*, there exists a probabilistic polynomial time algorithm S, the simulator, such that for all $x \in \mathcal{L}, S(x)$, transcript of the simulator, is "indistinguishable" from $View_V^*[P(x), V^*(x,z)]$. Here, View $_{V^*}[P(x), V^*(x,z)]$ denotes the distribution over all messages sent from P to V^* and randomness used by V^*. "Indistinguishable" can mean perfect zero knowledge, statistical zero knowledge or computational zero knowledge.

Honest verifier zero knowledge indicates if both the prover and verifier are honest, and there exists an efficient simulator that on the statement x, outputs a simulated accepting transcript with the same distribution as those of a real interaction.

$\Sigma-$Protocols A 3-round public coin interactive proof is called a Σ-protocol. The three messages sent in the protocol is usually denoted as (a, c, z) where a and z are messages from the prover and c is the challenge message sent by the verifier.

$\Sigma-$protocols are special honest-verifier zero-knowledge [9]: There exists a polynomial time simulator M, which can output an accepting transcript (a, c, z) with the same probability

distribution as interactions between P and V on input x when given the input x and a random challenge c.

A $\Sigma-$protocol is *k-special sound* if an efficient algorithm can recover the witness of the proof upon receiving the statement x and accepting transcripts (a, c_1, z_1), (a, c_2, z_2), ..., (a, c_k, z_k).

Fiat-Shamir transform Fiat-Shamir transform[11] is particularly useful in conjunction with $\Sigma -$ Protocols to generate a non-interactive proof. Instead of the verifier generating random messages and send then to the prover, the prover simulates the random message by hashing the transcript of the proof proceeding it. Then continue with the rest of the proof. In the end, the prover generates the entire transcript and lets any verifier to verify the validity of the proof. Later, the protocol is generalized to multi-round proof systems [1]. Fiat-Shamir transform and the generalization created a standard way to compile a public coin argument systems to non-interactive systems.

Schnorr's protocol [17] Schnorr's protocol is a classic sigma protocol, aiming to prove knowledge of a discrete log in a group G of prime order q. Let $g \neq 1$, and $x = g^w$ be public and the prover has the witness w, so the prover needs to prove that he knows w such that $x = g^w$. The protocol goes as the following:

1. The prover chooses z at random and sends $a = g^z$ to the verifier V.
2. The verifier chooses c at random and sends it to P.
3. Finally, P sends $r = (z + cw) \mathrm{mod} q$ to V and V checks that $g^r = ax^c$.

zk-SNARKs Zk-SNARKs, which stands for Zero-Knowledge Succinct Non-Interactive Argument of Knowledge, can be realized in many promising approaches, such as interactive proofs(IPs), probabilistic checkable proofs(PCPs), multi-prover interactive proofs(MIPs), or linear PCPs. They are zero knowledge proofs that are succinct and quick to verify, but usually at the expense of heavy computation on the prover side. All of these, aside from linear PCPs, proof systems, by applying a polynomial commitment scheme, can generate succinct proofs. Then apply Fiat-Shamir Transform so that the proof is non-interactive. Zk-SNRAKs have a wide range of applications. In particular, it succinctness and quick verification process make it compatible with blockchain applications. ZCash uses makes use of zk-SNARKs: The spender, receiver, and the amount of transaction are all hidden, but a short SNARK proof makes it easy for anyone to verify the validity of the statement.

Proofs of partial knowledge Proofs of partial knowledge allow a prover to convince a verifier that he knows the witness of one statement out of n statements without revealing which one he knows. This scenario can generalize to proving k-out-of-n statements: the prover convinces the verifier that he knows k out of n secrets.

The first protocol for proving one-out-of-n or k-out-of-n partial knowledge is by Cramer et al. [8]. The main idea of the protocol is of the following: the prover prepares the first round message of the statements that he knows the secrets for and simulates the full 3-round proof transcript of those that he does not know the secrets for. Then he sends the first round messages to the verifier. The verifier sends a random string. Finally, the prover splits the random string and use each part to produce the proof for the statements where he know the secrets. Along with the third round messages for the rest of the statement that he already prepared, he sends the third round messages to the verifier. The restriction is that the random string can only be split in a limited number of ways such that the prover must use random strings for the statements whose secrets he claims to know,

1.2 Commit-and-Prove Zero Knowledge Proof Systems

In [6], Campanelli et al. comes up with a new way to prove the accumulator verification in SNARK, which is based on a novel combination of sigma protocols, succinct proof of knowledge of exponent, and zkSNARKs for integer arithmetic.

Commit-and-Prove

1.3 RSA accumulator

RSA accumulators [5, 14] are used to prove set memberships from a prover to a verifier. The list of elements in the set is all odd primes. The core procedures include aggregate, prove membership and verify membership. As opposed to Merkle tree that has size logarithmic for set membership proofs, RSA accumulators have constant size proof. But problems for such protocols is that they require the numbers being proven are prime numbers.

- Setup(λ) Generate a group of unknown order and a generator for the group. $\mathbb{G} \xleftarrow{\$} GGen(\lambda)$, then $g \xleftarrow{\$} \mathbb{G}$.
- Aggregate(S) Return $A = g^{\prod_{s \in S} s}$.
- ProveMembership(S, x) Membership Proof for an element is calculating the accumulator without the aggregated item. Return $w = g^{\prod_{s \in S, s \neq x} s}$.
- VerifyMembership(A, w, x) Return 1 of $w^x = A$ Otherwise return 0.
- ProveBatchMembership($S, x_1, x_2, ..., x_n$) Set the subset $c = x_1, x_2, ..., x_n$. Return $w = g^{\prod_{s \in S, s \notin c} s}$.
- VerifyBatchMembership($A, S, w, x_1, x_2, ..., x_n$). Compute $x = \prod x_i$. Return 1 if $w^x = A$. Otherwise return 0.

1.4 State-of-the-art membership proofs

The typical approach to zero knowledge membership proofs is zkSNARKs based on Merkle Trees. A Merkle tree is a cryptographic primitive that after initial processing by iteratively calculating the hashes, enables one to prove the membership of an element at any time. However, to generate Merkle tree proof in SNARK results in large computation overhead comparing to proof in plaintext. For a set of n elements, the circuit has to encode $logn$ hash computations in the zkSNARK system. Furthermore, Merkle trees do not support batch membership proofs. for a batch of size m, $O(mlogn)$ hash computations are required, using Merkle Tree structure means repeating the proof for each element of the batch plus proving all elements are distinct, causing worse runtime for the prover.

There is no scalable solution for proving batch membership in zero-knowledge. We would like to consider batch memebership proofs with RSA accumulators. Although the naïve implementation still requires $O(m)$ RSA group operations ($1.8\dot{m}$ millions constraints [6]). In [6], Campenelli et al. proposed new techniques to use zkSNARKs with RSA accumulators.

Commit and Prove

Commit-and-prove zero knowledge proofs (CP-ZKPs) [13, 7] are a class of zero knowledge proofs where the prover proves the validity of a statement that involves some committed values. The informal definition is as follows: $\mathcal{L}_{\text{ck}} = \{c_x : Comm(x) = c_x \, , x \in \mathcal{L}\}$. One benefit of CP-ZKPs that is relevant to our problem is the property of *interoperability*. In another word, one can prove two different statements about the same value being committed using different zero knowledge proof systems. Of course, one can combine the commitment opening and other proof(s) as one large proof by putting them in a large circuit, but the approach is highly inefficient.

This motivated a proposal[2] to formalize the definitions of CP-ZKPs and have a framework for building CP-ZKPs, which involves instantiating a commitment scheme. What is particularly relevant from the proposal is the definition of CP-NIZKs (Commit and prove non-interactive zero knowledge proofs), which applies to the protocol of [6] detailed in 1.5.

Definition 1 *(CP-NIZK). Let $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of relations R over $\mathcal{D}_x \times \mathcal{D}_u \times \mathcal{D}_\omega$ such that $\mathcal{D}_u$ splits over ℓ arbitrary domains $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ for some arity parameter $\ell \geq 1$. Let Com = (Setup, Commit, VerCommit) be a commitment scheme (as per Definition 4.2) whose input space $\mathcal{D}$ is such that $\mathcal{D}_i \subset \mathcal{D}$ for all $i \in [\ell]$. A commit-and-prove NIZK for Com and $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ is a NIZK for a family of relations $\left\{\mathcal{R}_\lambda^{Com}\right\}_{\lambda\in\mathbb{N}}$ such that: - every $\mathbf{R} \in \mathcal{R}^{Com}$ is represented by a pair (ck, R) where $\text{ck} \in \text{Setup}\left(1^\lambda\right)$ and $R \in \mathcal{R}_\lambda$; - $\mathbf{R}$ is over pairs $(\mathbf{x}, \mathbf{w})$ where the statement is $\mathbf{x} := \left(x, (c_j)_{j\in[\ell]}\right) \in \mathcal{D}_x \times \mathcal{C}^\ell$, the witness is $\mathbf{w} := \left((u_j)_{j\in[\ell]}, (o_j)_{j\in[\ell]}, \omega\right) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell \times \mathcal{O}^\ell \times \mathcal{D}_\omega$, and the relation $\mathbf{R}$ holds iff*

$$\bigwedge_{j\in[\ell]} \text{VerCommit}\left(\text{ck}, c_j, u_j, o_j\right) = 1 \wedge R\left(x, (u_j)_{j\in[\ell]}, \omega\right) = 1$$

Furthermore, when we say that CP *is knowledge-sound for a relation generator* $\mathcal{RG}$ *and auxiliary input generator* $\mathcal{Z}$ *(denoted* $\mathrm{KSND}(\mathcal{RG}, \mathcal{Z})$*, for short) we mean it is a knowledge-sound NIZK for the relation generator* $\mathcal{RG}_{\mathrm{Com}}\left(1^{\lambda}\right)$ *that runs* $\mathrm{ck} \leftarrow \mathrm{Setup}\left(1^{\lambda}\right)$ *and* $(R, \mathrm{aux}_R) \leftarrow \mathcal{RG}\left(1^{\lambda}\right)$*, and returns* $((\mathrm{ck}, R), \mathrm{aux}_R)$*.*

1.5 New zkSNARK for RSA accumulators [6]

Campanelli et al.[6] designed a succinct zero knowledge proof of set membership for a batch of elements, which sought to improve the scalability challenge of the problem. The main scheme is a $\Sigma-$protocol along with several SNARKs to assist with the proof. Notably, no RSA group operations is encoded in the SNARK system. Both proof size and verification time is $O(1)$.

Blockchain transactions can benefit from such zero knowledge proofs. When new transactions are broadcasted, validators run validity checks: checking if the new transactions comply with the global state. In bitcoins, we have UTXO; in ethereum, we have account balance pairs. Ideally, verification for large batches should not require the verifiers to store all global states. We observe Monero could also benefit from such constructions in their ring signature protocol, which will be discussed more in detail 2.2

First approach

We first show a naive approach of the problem that employs a $\Sigma-$protocol. Given a random element g from a group of unknown group order, $S = x_1, x_2, ..., x_n$ is the set of elements to be accumulated assuming all elements are prime, $\mathrm{acc} = g^{\prod_{x_i \in S} x_i}$. $S' = u_1, ..., u_m$. $W = g^{\prod_{s \in S, s \notin S'} s}$.The prover aims to prove that $S' \in S$.

1. The prover chooses a large random r, and computes $R = W^r$ and sends it to the verifier.
2. The verifier chooses c at random and sends it to the prover.
3. The prover sends the verifier $k = r + c\dot{\prod}_{u_i \in S'} u_i$. The verifier checks if $R \cdot \mathrm{acc}^c = W^k$.

Problems The main problem with this sigma protocol is that it does not fully hide the elements in the batch to be proved: S'. As the verifier needs to know W, which leak information about which elements are in the batch, especially if the batch size is small. The verifier could test all potential elements or element combinations by brute force.

The $\Sigma-$protocol only provides proof for the elements in the batch, but fails to link the elements with their commitments.

The proof is not succinct as the last message from the prover to verifier is O(m) bits long. It is not of a constant size.

Solutions [6] uses randomization techniques to hide the elements in the batch by adding additional small prime numbers, so that the witness is no longer the aggregation of elements from the original set. Because of the introduction of small primes, a few SNARKs are needed to prove that the numbers is the batch are not the newly introduced small primes, which we will expand later.

The prover uses a zkSNARK to link the commitments to every element in S' to the $\Sigma -$ protocol. Specifically, it connects the commitment $c_{\vec{u}}$ with the last message of the $\Sigma -$ protocol since that is the only place u_i's are included.

Finally, because the last message from the prover is proportional to the size of S', [6] decides to use of a proof of knowledge of an exponent(PoKE) [4]. With PoKE, the prover does not send $k = r + c\dot{\prod}_{u_i \in S'} u_i$ directly, but sends a zkSNARK to prove the knowledge of k for the relation $(\hat{\text{acc}}^c R) = \left(\hat{W}_{\vec{u}}\right)^k$ where $\hat{\text{acc}}^c$ and $\hat{W}$ correspond to the previous acc and W after adding small primes.

New Sigma Protocol

We discuss the new protocol that is computationally witness hiding. The new technique relies on adding small prime numbers to the accumulator.

Preprocessing

Let $\mathbb{P}_n = \{2, 3, 5, 7, \ldots, p_n\}$ be the set of the smallest n prime numbers.

First, the prover and verifier modify the accumulator acc to contain the smallest 2λ primes by computing

- $\hat{\text{acc}} \leftarrow \text{acc}^{(\prod_{p_i \in \mathbb{P}_{2\lambda}} p_i)}$. This is the same as $\text{Aggregate}(S \cup \mathbb{P}_{2\lambda})$.

Second, the prover will randomly chose a subset of $\mathbb{P}_{2\lambda}$ to be added to the batch to be proven.

- The prover randomly samples 2λ bits $b_1, \ldots, b_{2\lambda} \xleftarrow{\$} \{0, 1\}$ and set $s = \prod_{p_i \in \mathbb{P}_{2\lambda}} p_i^{b_i}$ and $\bar{s} := \prod_{p_i \in \mathbb{P}_{2\lambda}} p_i^{1-b_i}$.
- The prover computes the complement of the batch to be proven and the selected small primes $\hat{W} \leftarrow W^{\bar{s}} = g_?^{(\prod_{x_i \in S \setminus X} x_i) \cdot \left(\prod_{p_i \in \mathbb{P}_{2\lambda}} p_i^{1-b_i}\right)}$.

By adding some small primes into the accumulator, essentially, the verifier will not be able to guess which elements of S are part of the batch; therefore, witness hiding was achieved.

After the preprocessing, the prover and verifier proceed with the $\Sigma-$Protocol detailed above.

Proof of computational indistinguishability

Now, we would like to prove $\hat{W}$ computed by the prover at the end of preprocssing is computationally indistinguishable from a random group element. First, we introduce the DDH-II assumption [10] and proves that under DDH-II, $\hat{W}$ is indistinguishable from random.

DDH-II Assumption There are variations of DDH-II assumptions for generic groups, prime order groups an unknown order groups, which are all derived from the original DDH assumption, and security has been proven for each category. We are using the unknown order group model, and the idea is the following: For a group $\mathbb{G}_?$ of unknown order and $g_?$,$g_?^a$,$g_?^b$, where a is drawn from a certain (not necessarily uniform) distribution, with sufficient min-entropy, whereas b is picked uniformly at random, $g_?^{ab}$ is indistinguishable from a uniform element in $\mathbb{G}_?$.

Formally, let $\mathbb{G}_? \leftarrow \mathcal{G}_? \left(1^\lambda\right)$ and $g_? \xleftarrow{\$} \mathbb{G}_?$. Let $\mathcal{WS}_{2\lambda}$ be a well-spread distribution with domain $\mathcal{X}_{2\lambda} \subseteq [1, \text{minord}(\mathbb{G}_?)]$. Then for any PPT $\mathcal{A}$:

$$\left|\Pr\left[\mathcal{A}\left(g_?^x, g_?^y, g_?^{xy}\right) = 0\right] - \Pr\left[\mathcal{A}\left(g_?^x, g_?^y, g_?^t\right) = 0\right]\right| = \text{negl}(\lambda)$$

where $x \xleftarrow{\$} \mathcal{WS}_{2\lambda}$ and $y, t \xleftarrow{\$} \left[1, \text{maxord}\left(\mathbb{G}_?\right) 2^\lambda\right]$.

In the RSA accumulator, $\mathcal{D}_{2\lambda}$ is well spread as there are $2^{2\lambda}$ distinct possible outcomes. Therefore, for each $\bar{s} = \prod_{p_i \in \mathbb{P}_{2\lambda}} p_i^{1-bi}$, $\Pr\left[\bar{s} \xleftarrow{\$} \mathcal{D}_{2\lambda}\right] = 1/2^{2\lambda}$ for every $\bar{s}$.

Security proof under DDH-II assumption For any parameters $\text{pp} \leftarrow \text{Setup}\left(1^\lambda\right)$, set S (where $S \cap \mathbb{P}_{2\lambda} = \emptyset$) , $R \xleftarrow{\$} \mathbb{G}_?$ and $\hat{W}$ computed as described above it holds:

$$|\Pr[\mathcal{A}(\text{pp}, S, \hat{W}) = 0] - \Pr[\mathcal{A}(\text{pp}, S, R) = 0]| = \text{negl}(\lambda)$$

for any $PPT\mathcal{A}$, under the DDH-II assumption for $\mathbb{G}_?$ and $\mathcal{D}_{2\lambda}$. Proof. Call $\mathcal{A}$ an adversary achieving a non-negligible advantage ϵ above, i.e. $\epsilon := |\Pr[\mathcal{A}(\text{pp}, S, \hat{W}) = 0] - \Pr[\mathcal{A}(\text{pp}, S, R) = 0]|$We construct an adversary $\mathcal{B}$ against DDH-II that, using adversary $\mathcal{A}$, gains the same advantage. $\mathcal{B}$ receives (G ? $, g_?, g_?^{\bar{s}}, g_?^r, g_?^{b\bar{s}r+(1-b)t}$), where $\bar{s} \leftarrow \$\mathcal{D}_{2\lambda}$ and $r, t \leftarrow \$\left[1, \text{maxord}(\mathbb{G}_?) 2^\lambda\right]$. Then it chooses arbitrarily an element u and sets $S = \{u\}, \text{pp} \leftarrow (\mathbb{G}_?, g_?^r)$ and $V = g_?^{b\bar{s}r+(1-b)t} \cdot \mathcal{B}$ sends (pp, S, V) to the adversary $\mathcal{A}$, who outputs a bit b^*. Finally, $\mathcal{B}$ outputs b^*.

First, notice that $g_?^r$ is statistically close to a random group element of $\mathbb{G}_?$, meaning that $\mathcal{A}$ cannot distinguish pp from parameters generated by Acc.Setup $\left(1^\lambda\right)$. Furthermore if

$b = 0$**:** V is again a (statistically indistinguishable element from a) uniformly random group element of $\mathbb{G}_?$ therefore $\Pr[\mathcal{B} = 0 \mid b = 0] = \Pr[\mathcal{A}(\text{pp}, S, R) = 0]$.

$b = 1$: $V = g_?^{r \cdot \bar{s}} = \hat{W}_u$ is a witness of u so $\Pr[\mathcal{B} = 0 \mid b = 1] = \Pr[\mathcal{A}(\text{pp}, S, \hat{W}) = 0]$.

In both circumstances, $\Pr[\mathcal{B} = 0]$ is the same as the probability of adversary $\mathcal{A}$ winning the game, which is negligible.

Proof of Knowledge of Exponent

Although the new Σ-protocol satisfies the desired zero knowledge property, the proof is not succinct as the last message grows with the size of the batch. Therefore, we use a recent result Boneh et al.[4] called Proof of Knowledge of Exponent(PoKE) to prove the relation

$$R^{\text{PoKE}}(u, w; x) = 1 \Leftrightarrow u^x = w$$

for a group of unknown order $\mathbb{G}_?$ and a random element from the group. Here, $u, w \in \mathbb{G}_?$ and x is an integer.

Params: $\mathbb{G}_? \xleftarrow{\$} GGen(\lambda)$; Inputs: $u, w \in \mathbb{G}$; Witness: $x \in \mathbb{Z}$; Claim: $u^x = w$

1. Verifier sends $\ell \xleftarrow{\$} \text{Primes}(\lambda)$.
2. Prover finds the quotient $q \in \mathbb{Z}$ and residue $r \in [\ell]$ such that $x = q\ell + r$. Prover sends $(Q = u^q, r)$ to the Verifier.
3. Verifier accepts if $r \in [\ell]$ and $Q^\ell u^r = w$.

To make the protocol non-interactive, we apply Fiat-Shamir transform: instead of l being sampled by the verifier, the prover applies a hash function on the input (crs, A, B) to sample a prime of size 2λ. Then sends the verifier the proof. The verification process is the same. Note this protocol is insecure if applied to a base freely chosen by the prover as in some circumstances the prover can select Q and r based on the information he has about u and w to still generate a valid proof without knowing x. However, this is sufficient for our case because we will combine it with a SNARK that proves the correct computation of r based on the commitments, i.e. $r = x \bmod l$. We will discuss this specific SNARK for integer arithmetic relations below.

The proof is succinct as the proof size and verification time are independent of the size of the exponent as $x \in \mathbb{Z}$ can be much larger than $|\mathbb{G}|$.

CP-SNARK for integer arithmetic relations

We need a snark proof for connecting the $\Sigma-$Protocol with the vector commitment of the batch. It is natural to connect the last message with the commitment to show the elements used in the proof are those being committed. To summarize, we are proving the relationship

$$R^{\text{modarithm}}\left(c_{\vec{u}}, c_{s,r}, h, \ell, \hat{k}\right) = 1 \Leftrightarrow \hat{k} = s \cdot h \cdot \prod_{i \in [m]} u_i + r \mod \ell$$

with a SNARK $\text{cp}\Pi^{\text{modarithm}}$. Here, $\vec{u}$ is a vector of integers in the batch and $c_{\vec{u}}$ is the commitment. $s = \prod_{p_i \in \mathbb{P}_{2\lambda}} p_i^{b_i}$ and r, randomly sampled in the first round of the $\Sigma-$Protocol are also integers committed. $l, h \in \mathbb{Z}$ and $\hat{k}$ is the final message of the $\Sigma-$Protocol.

The relationship can be encoded in a SNARK in the format of

$$R^{\text{arithmetic}}\left(c_{\vec{u}}, c_{s,r}, h, \ell, \hat{k}; q\right) = 1 \Leftrightarrow q\ell + \hat{k} = s \cdot h \prod_i u_i + r$$

where q is the witness.

CP-SNARK for inequalities

Because of the added small primes that could be confused as elements in the batch, the prover needs to show every element in the batch is bigger than B, a pubic parameter. We use a SNARK $\text{cp}\,\Pi^{\text{bound}}$ to prove this relation:

$$R^{\text{bound}}\ (c_{\vec{u}}, B) = 1 \Leftrightarrow \forall i, u_i > B$$

Hashing integers to primes

So far, we've been assuming the input to the RSA accumulator are large primes. However, we would like to expand the functionality to accept arbitrary elements. The main idea for the adaptation is using collision-resistant hash functions that map arbitrary elements to prime numbers and prove the correct mapping. In [6], they used the two mapping algorithms from [3]. Assume a collision-resistant function $\text{H} : \{0,1\}^\eta \times \{0,1\}^\iota \to \{0,1\}^{\mu-1}$ that inputs and outputs binary numbers. Define the function $\text{H}_{\text{prime}} : \{0,1\}^\eta \to \text{Primes}\,(2^{\mu-1}, 2^\mu)$ that looks for the first $j \in [0, 2^\iota - 1]$, s.t. the integer representation of the binary string $1|\text{H}(u,j)$ is a prime. The function will fail with negligible probability. Now we discuss the implementation of H. [3] gives two candidates:

1. Pseudorandom function Let $\text{H}(u,j) := \text{F}_\kappa(u,j)$ where $\text{F}_\kappa : \{0,1\}^{\eta+\iota}$ is a PRF with public seed κ and $\iota = |\log \mu\lambda|$. The expected runtime is $O(\mu)$ for the density of primes and the resulting $\text{H}_{\text{prime}}(u)$ fails with probability at most $exp(-\lambda)$.

2. Deterministic map Let $f(u) := 2(u+2)\log_2(u+1)^2$ and set $\text{H}(u,j) := f(u) + j$ with $u > 2^{\eta-1}$ and $j \in (f(u), f(u+1))$. If $\mu > \eta$, the function is collision free. This complies with Cramer's conjecture that implies $(f(u), f(u+1))$ contains a prime when u is sufficiently large.

Using either of these two algorithms gives a satisfactory mapping function $\text{H}_{\text{prime}}(u)$. However, by sending arbitrary elements to primes and only using these primes in commitments and RSA accumulators, we need to add proofs for the mapping by utilizing the commit-and-prove modularity. Let $(\hat{u_1}, \hat{u_2}, ..., \hat{u_m})$ by the original elements that map to $(u_1, u_2, ..., u_m)$, i.e. $u_i = \text{H}_{\text{prime}}(\hat{u_i})$. We use a CP-SNARK, $\text{CP}_{\text{HashEq}}$ to prove that $\forall i, u_i = \text{H}_{\text{prime}}(\hat{u_i})$. The

relation to be proved (We call it $R_{\text{HashEq}} : \{0,1\}^{\mu} \times \{0,1\}^{\eta} \times \{0,1\}^{\iota}$) can be written as the following:

$$R_{\text{HashEq}}(u_1, u_2, \omega) = 1 \iff u_1 = (1|\text{H}(u_2, \omega))$$

Here, u_1 refers to the resulting prime, u_2 refers to the original element, and ω is the index used to generate the prime, i.e first number in $[0, 2^{\iota} - 1]$ that makes $(1|\text{H}(u_2, \omega))$ a prime. Note that in the proof, we skip the iteration process of finding j, so *only* one hash computation is encoded, reducing the complexity of the circuit.

Full Protocol

To summarize, the full protocol contains a $\Sigma-$Protocol and several zkSNARKs to support the $\Sigma-$Protocol. It has the following setting: a prover wants to prove membership of elements in an accumulator under zero knowledge. The elements are committed and each element is a prime number greater than the 2λ-th prime. Since both parties hold acc, the accumulated value of entire set, by property of RSA accumulator, the prover wants to show it has $W_{\vec{u}}$ such that $W_{\vec{u}}^{\prod_i u_i} = \text{acc}$ for a batch of elements $\vec{u} = (u_1, \ldots, u_m)$.

The core of the proof is a novel $\Sigma-$Protocol: by injecting the first 2λ primes into the accumulator, the prover prevents the verifier from guessing the elements in the batch. As a result of that, the prover needs to prove no element in the batch is a small prime($\text{cp}\,\Pi^{\text{bound}}$). The last step of the $\Sigma-$Protocol produces a message of large size. To preserve succinctness of the proof, the prover employs a PoKE proof along with a SNARK that connects the $\Sigma-$Protocol with the commitment($\text{cp}\Pi^{\text{modarithm}}$). The following is a discription of the protocol:

Setup$\left(1^{\lambda}, \text{ck}, \text{pp}\right)$**:**

- $\text{crs}_2 \leftarrow \text{cp}\Pi^{\text{modarithm}}.\text{Setup}\left(1^{\lambda}, \text{ck}, R^{\text{modarithm}}\right)$
- $\text{crs}_3 \leftarrow \text{cp}\Pi^{\text{bound}}.\text{Setup}\left(1^{\lambda}, \text{ck}, R^{\text{bound}}\right)$
- return crs :=(ck, pp, $\text{crs}_{\text{arithmetic}}$, $\text{crs}_{\text{bound}}$)

Prove$(\text{crs}, \text{acc}, c_{\vec{u}}; W_{\vec{u}}, \vec{u}, o_{\vec{u}})$**:**

- $\hat{\text{acc}} \leftarrow \text{acc}^{\prod_{p_i \in \mathbb{P}_{2\lambda}} p_i}$
- Let $u^* = \prod_i u_i, p^* = \prod_{p_i \in \mathbb{P}_{2\lambda}} p_i$
- Sample $b_1, \ldots, b_{2\lambda} \xleftarrow{\$} \{0,1\}$. Let $s := \prod_{p_i \in \mathbb{P}_{2\lambda}} p_i^{b_i}, \bar{s} := \prod_{p_i \in \mathbb{P}_{2\lambda}} p_i^{1-b_i}$
- $\hat{W}_{\vec{u}} \leftarrow W_{\vec{u}}^{\bar{s}}$
- Sample $r \xleftarrow{\$} \{0,1\}^{\|p^*\|+\|u^*\|+2\lambda}$. Then compute $c_{s,r} \leftarrow \text{Comm}_{\text{ck}}(s, r; o_{s,r})$ and $R \leftarrow \hat{W}_{\vec{u}}^{r}$.

- Compute challenge message $h \leftarrow H\left(\mathrm{crs} \,\|\, \mathrm{acc} \,\|c_{\vec{u}} \,\|c_{s,r}\|\, \hat{W}_{\vec{u}} \| R\right)$ and the last message of Σ-Protocol $k \leftarrow r + (u^* s)\, h$
- $\ell \leftarrow H_{\mathrm{prime}}\left(\left(\mathbb{G}_?, g_?\right), \hat{W}_{\vec{u}},\ \hat{acc}^h R\right)$
- $\pi_1 \leftarrow \Pi^{\mathrm{PoKE}} \cdot \mathrm{Prv}\left(\left(\mathbb{G}_?, g_?\right), \hat{W}_{\vec{u}}, \hat{acc}^h R; k\right)$. Parse π_1 as $(Q, \hat{k})$.
- $\pi_2 \leftarrow \mathrm{cp}\Pi^{\mathrm{modarithm}} \cdot \mathrm{Prv}\left(\mathrm{crs}_2, c_{\vec{u}}, c_{s,r}, h, \ell, \hat{k}; \vec{u}, o_{\vec{u}}, r, s, o_{s,r}\right)$
- $\pi_3 \leftarrow \mathrm{cp}\,\Pi^{\mathrm{bound}} \cdot \mathrm{Prv}\left(\mathrm{crs}_3, c_{\vec{u}}, p_{2\lambda}; \vec{u}, o_{\vec{u}}\right)$
- return $\pi = \left(\hat{W}_{\vec{u}}, R, c_{s,r}, \pi_1, \pi_2, \pi_3\right)$

Verify$(\mathrm{crs}, \mathrm{acc}, c_{\vec{u}}, \pi)$**:**

- $\hat{\mathrm{acc}} \leftarrow \mathrm{acc} \prod_{p_i \in \mathbb{P}_{2\lambda}} p_i$
- Parse π as $\left(\hat{W}_{\vec{u}}, R, c_{s,r}, \pi_1, \pi_2, \pi_3\right)$ and π_1 as $(Q, \hat{k})$
- $\ell \leftarrow H_{\mathrm{prime}}\left(\left(\mathbb{G}_?, g_?\right), \hat{W}_{\vec{u}},\ \hat{acc}^h R\right)$
- $h \leftarrow H\left(\mathrm{crs} \,\|\, \mathrm{acc} \,\|c_{\vec{u}} \,\|c_{s,r}\|\, \hat{W}_{\vec{u}} \| R\right)$
- Reject if Π^{PoKE} . $\mathrm{Vfy}(\mathbb{G}_?, g_?), \hat{W}_{\vec{u}},\ \hat{acc}^h R, \pi_1) \neq 1$
- Reject if $\mathrm{cp}\,\Pi^{\mathrm{modarithm}}$.$\mathrm{Vfy}\left(\mathrm{crs}_2, c_{\vec{u}}, c_{s,r}, h, \ell, \hat{k}, \pi_2\right) \neq 1$
- Reject if $\mathrm{cp}\,\Pi^{\mathrm{bound}}$.$\mathrm{Vfy}\left(\mathrm{crs}_3, c_{\vec{u}}, p_{2\lambda}, \pi_3\right) \neq 1$

Chapter 2

New Ring Signature Construction

2.1 Ring Signatures

Ring signatures are a type of digital signatures that can hide the identity of the signer within a group by Rivest et al [16]. Ring signatures make it possible to specify a set of possible signers and it is publicly verifiable that the signature is produced by one of the members but one cannot identify the member who creates the signature. Group signatures offer a similar guarantee, but always have one or more group leaders who are in charge of adding members and revealing the identity of the signer afterwards when a dispute arises. Therefore, a group signature is useful when there is a prearranged group of users who would like to cooperate.

However, ring signatures do not have a prearranged group of users and anonymity of the actual signer is never revoked. Adding or deleting members can be done by the signer.

Definition 2 (Ring Signatures) *A set of possible signers is called a* ring*, and the ring member who produces the signature is called a* signer*. A ring signature scheme is defined by the following PPT algorithms:*

KeyGen(1^λ)*: takes in the security parameter* 1^λ*, and outputs a pair of public and secret keys* (vk, sk)*.*

Sign(sk, m, R)*: takes in a secret key* sk*, a message* m*, and a list of public keys, which are also called verification keys,* $R = vk_1, ..., vk_r$ *and outputs a signature* σ*.*

Verify(m, σ, R) *takes in a message* m*, its signature* σ*, and a list of verification keys* $R = vk_1, ..., vk_r$*. Output either true or false.*

Properties of Ring Signatures

Correctness: We require that a user can sign any message on behalf of a ring where she is a member. A ring signature scheme (Setup, KGen, Sign, Vfy) has perfect correctness if for all adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{ll} pp \leftarrow \text{Setup}\left(1^\lambda\right); (vk, sk) \leftarrow \text{KGen}(pp) & \text{Vfy}_{pp}(M, R, \sigma) = 1 \\ (M, R) \leftarrow \mathcal{A}(pp, vk, sk); \sigma \leftarrow \text{Sign}_{pp,sk}(M, R) & \text{or } vk \notin R \end{array}\right] = 1$$

Set-up free: The signer does not need confirmation from the other ring members when adding them to the ring. The signer only needs to know the public keys of all ring members.

Anonymity: A signature should not leak any information about the identity of the signer from a ring. We say that a ring signature scheme RS = (KeyGen, Sign, Verify) is anonymous against full key exposure, if for every $q = \text{poly}(\lambda)$ and every PPT adversary $\mathcal{A}$ it holds that $\mathcal{A}$ has at most negligible advantage in the following experiment. $\text{Exp}_{\text{RS-Anon}}(\mathcal{A})$:

1. For all $i = 1, \ldots,$ q the experiment generates the keypairs $(\text{VK}_i, \text{SK}_i) \leftarrow \text{KeyGen}\left(1^\lambda, r_i\right)$ using random coins r_i and sends $\text{VK}_1, \ldots, \text{VK}_q$ and $\text{r}_1, \ldots, \text{r}_q$ to $\mathcal{A}$.
2. The adversary provides a challenge $(\text{R}, \text{m}, i_0, i_1)$ to the experiment, such that VK_{i_0} and VK_{i_1} are in the ring R. The experiment flips a random bit $b \leftarrow \$\{0,1\}$, computes $\Sigma^* \leftarrow \text{Sign}\left(\text{sK}_{i_b},\ \text{m}, \text{R}\right)$ and outputs Σ^* to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a guess b'. If $b' = b$, the experiment outputs 1 , otherwise 0

The advantage of $\mathcal{A}$ is defined by $\text{Adv}_{\text{RS}-\text{Anon}}(\mathcal{A}) = \left|\Pr\left[\text{Exp}_{\text{RS}-\text{Anon}}(\mathcal{A}) = 1\right] - \frac{1}{2}\right|$.

Unforgeability: $\text{Exp}_{\text{RS-Unf}}(\mathcal{A})$:

1. For all $i = 1, \ldots, q$ the experiment generates the keypairs $(\text{VK}_i, \text{SK}_i) \leftarrow \text{RS.KeyGen}\left(1^\lambda, \text{r}_i\right)$ using random coins r_i. It sets $\mathcal{VK} = \{\text{VK}_1, \ldots, \text{VK}_q\}$ and initializes a set $\mathcal{C} = \emptyset$.
2. The experiment provides $\text{VK}_1, \ldots, \text{VK}_q$ to $\mathcal{A}$.
3. $\mathcal{A}$ is now allowed to make the following queries: $(\text{sign}, i,\ \text{m}, \text{R})$: Upon a signing query, the experiment checks if $\text{VK}_i \in \text{R}$, and if so computes $\Sigma \leftarrow \text{RS. Sign}\left(\text{SK}_i,\ \text{m}, \text{R}\right)$ and returns Σ to $\mathcal{A}$. Moreover, the experiment keeps a list of all signing queries. (corrupt, i) : Upon a corruption query, the experiment adds VK_i to $\mathcal{C}$ and returns r_i to $\mathcal{A}$.
4. In the end, $\mathcal{A}$ outputs a tuple $(\text{R}^*,\ \text{m}^*, \Sigma^*)$. If it holds that $\text{R}^* \subseteq \mathcal{VK} \backslash \mathcal{C}$ (i.e. none of the keys in R^* were corrupted), $\mathcal{A}$ never made a signingquery of the form $(\text{sign}, \cdot, \text{m}^*, \text{R}^*)$ and it holds that
$$\text{Verify}\left(\text{R}^*,\ \text{m}^*, \Sigma^*\right) = 1$$
then the experiment outputs 1 , otherwise 0 .

The advantage of $\mathcal{A}$ is defined by $\text{Adv}_{\text{RS}-\text{Unf}}(\mathcal{A}) = \Pr\left[\text{Exp}_{\text{RS}-Unf}(\mathcal{A}) = 1\right]$.

Applications of Ring Signatures

Ring Signatures are used in e-commerce, e-voting. And notably, they are adopted by Monero, a cryptocurrency, to provide user privacy. Monero modified the classic definiton of ring signatures to cater to their application. Specifically, they created linkable ring signatures, and by incorporating all coins of a user, the technique was named RingCT2.2.

2.2 RingCT of Monero

Monero is a decentralized anonymous cryptocurrency. Originally, it was based on CryptoNote, which uses ring signatures and one-time keys to hide the source and receiver of transactions. To further improve the protocol, Monero implements hidden amounts for a transaction, which is based on Conidential Transactions on a side-chain in Bitcoin. RingCT [15], short for Ring Confidential Transactions, is how transaction amounts are hidden in Monero. In summary, a Multilayered Linkable Spontaneous Anonymous Group Signature (MLSAG) will be used to combine Confidential Transanctions and ring signatures in order to achieve anonymity of sender and receiver, infeasibility of double-spending, and enabling of multiple inputs and outputs.

2.3 Compile zero knowledge proof into ring signatures

Naturally, we can compile a zero knowledge proof in the format of sigma protocols to ring signatures by applying Fiat-Shamir transform. The challenge message of the $\Sigma-$protocol is generated by computing the hash of the first message *plus* the message to be signed, we transform the zero knowledge proof system to ring signatures.

2.4 Ring signatures from commit-and-proof zero knowledge proofs

Previous works of ring signatures have circuit size and hence signature size grow linearly in the size of the rings $|R|$. Alternatively, by using a Merkle tree, the size if now logarithmic in $|R|$. The main appeal of using zero knowledge proofs for set membership is the signature size does not grow with the size of the ring. It is constant or proportional to the size of the batch when multiple elements are involved in the proof.

As a result of properties of a commit-and-proof protocol, a prover can append any additional proof of the committed value, including discrete log or pseudo-random function computation, which are often used as public and private key generations.

Bibliography

[1] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Proceedings, Part II, of the 14th International Conference on Theory of Cryptography - Volume 9986*. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 31–60. ISBN: 9783662536438. DOI: 10.1007/978-3-662-53644-5_2. URL: https://doi.org/10.1007/978-3-662-53644-5_2.

[2] Daniel Benarroch et al. "Proposal: Commit-and-Prove Zero-Knowledge Proof Systems and Extensions". In: 2021.

[3] Daniel Benarroch et al. "Zero-Knowledge Proofs for Set Membership: Efficient, Succinct, Modular". In: *IACR Cryptol. ePrint Arch.* 2019.

[4] Dan Boneh, Benedikt Bünz, and Ben Fisch. "Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains". In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 1188.

[5] Jan Camenisch and Anna Lysyanskaya. "Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials". In: *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*. Vol. 2442. Lecture Notes in Computer Science. Springer, 2002, pp. 61–76. DOI: 10.1007/3-540-45708-9_5. URL: https://iacr.org/archive/crypto2002/24420061/24420061.pdf.

[6] Matteo Campanelli et al. *Succinct Zero-Knowledge Batch Proofs for Set Accumulators*. Cryptology ePrint Archive, Report 2021/1672. https://ia.cr/2021/1672. 2021.

[7] Ran Canetti et al. "Universally composable two-party and multi-party secure computation". In: *STOC '02*. 2002.

[8] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols". In: *CRYPTO*. 1994.

[9] Ivan Damgård. "On Σ-protocols". In: *Lecture Notes, University of Aarhus, Department for Computer Science* (2002), p. 84.

[10] Ivan Damgård, Carmit Hazay, and Angela Zottarel. "On the Generic Hardness of DDH-II". In: 2012.

[11] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: Berlin, Heidelberg: Springer-Verlag, 1987. ISBN: 0387180478.

[12] S Goldwasser, S Micali, and C Rackoff. "The Knowledge Complexity of Interactive Proof-Systems". In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. STOC '85. Providence, Rhode Island, USA: Association for Computing Machinery, 1985, pp. 291–304. ISBN: 0897911512. DOI: 10.1145/22145.22178. URL: https://doi.org/10.1145/22145.22178.

[13] Joe Kilian. "Uses of randomness in algorithms and protocols". In: 1990.

[14] Helger Lipmaa. "Secure Accumulators from Euclidean Rings without Trusted Setup". In: *Applied Cryptography and Network Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 224–240. ISBN: 978-3-642-31284-7.

[15] Shen Noether, Adam Mackenzie, and the Monero Research Lab. "Ring Confidential Transactions". In: *Ledger* (2016), pp. 1–18.

[16] Ronald L. Rivest, Adi Shamir, and Yael Tauman Kalai. "How to Leak a Secret". In: *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*. Vol. 2248. Lecture Notes in Computer Science. Springer, 2001, pp. 552–565. DOI: 10.1007/3-540-45682-1_32. URL: https://iacr.org/archive/asiacrypt2001/22480554.pdf.

[17] Claus Schnorr. "Efficient signature generation by smart cards". In: *Journal of Cryptology* 4 (Jan. 1991), pp. 161–174. DOI: 10.1007/BF00196725.