Batching non-membership proofs with bilinear accumulators

Steve Thakur

Axoni

Abstract

In this short paper, we provide protocols to batch and aggregate multiple non-membership proofs into a single proof of constant size with bilinear accumulators. We subsequently use the accumulator to construct a bilinear Vector Commitment with constant sized openings and a linear public parameter. Furthermore, we have designed the protocols so that the Verifier needs a constant amount of storage for verification despite the linear public parameter. We also provide ways to speed up the verification of membership and non-membership proofs and to shift most of the computational burden from the Verifier to the Prover. Since all the protocols are public coin, they can be made non-interactive with a Fiat-Shamir heuristic.

1 Introduction

A cryptographic *accumulator* is a one-way membership function that answers a query about whether a potential set of candidates is a subset of a given set without revealing the individual members of the set. In this paper, we study a class of asymmetric accumulators that is based on bilinear pairings of cyclic groups of prime order. First introduced by Nguyen in [Ngu05], these accumulators have the major advantage over the better known accumulator of a Merkle tree in that membership proofs are of constant size and multiple membership proofs can be batched or aggregated together into a single proof. Furthermore, it was shown in [DT08] that they also allow for non-membership witnesses. These advantages are shared with accumulators based on groups on unknown order, of which the two best known types are RSA accumulators and (imaginary quadratic) class group accumulators.

A Vector Commitment (VC) is a closely related primitive [CF13]. It provides the same functionality as an accumulator, but for an ordered list of elements rather than a set. A VC is a position binding commitment and can be opened at any position to a unique value with a short proof (preferrably independent of the length of the vector). Subvector commitments [LM18] are VCs for which a subset of the vector positions can be opened in a single short proof.

Since bilinear accumulators require groups far smaller than RSA groups for the same level of security, we expect them to be substantially faster than RSA accumulators when it comes to accumulation, generation of membership proofs (witnesses) and verification. In recent years, cryptographic accumulators have seen a growing interest as a potential alternative to Merkle trees for blockchains. In particular, significant progress was made in the paper [BBF19] where the authors constructed protocols for constant sized non-membership proofs for arbitrarily large sets of data elements in accumulators based on groups of unknown order. In this paper, we show that this is also possible for bilinear accumulators and subsequently construct a bilinear Vector Commitment with constant sized openings and a linear public parameter. The protocols are designed so as to allow the Verifier to maintain constant sized storage despite the linear public parameter.

Furthermore, we adapt techniques from [BBF19] and [Wes18] to speed up verifications of membership and non-membership proofs and to shift most of the computational and storage burden from the Verifier to the Prover. In particular, we provide a protocol to reduce the Verifier's task of verifying membership and non-membership proofs to a constant number of group operations independent of the number of data elements to be batched.

1.1 Notations and terminology

As usual, $\mathbb{F}_q$ denotes the finite field with q elements for a prime power q and $\overline{\mathbb{F}}_q$ denotes its algebraic closure. $\mathbb{F}_q^*$ denotes the cyclic multiplicative group of the non-zero elements of $\mathbb{F}_q$. For univariate polynomials $f(X), g(X) \in \mathbb{F}_q[X]$, we denote by $\gcd(f(X), g(X))$ the unique *monic* polynomial $h(X) \in \mathbb{F}_q[X]$ that generates the principal ideal of $\mathbb{F}_q[X]$ generated by $f(X)$ and $g(X)$.

Definition 1.1. Batching and aggregation: Following the terminology of [BBF19], we use the term *batching* for the action of creating a single membership (or non-membership) witness for multiple data elements. *Aggregation* refers to the action of creating a single membership or non-membership proof for the data elements using individual witnesses that have already been created.

Neither of these mechanisms is afforded by Merkle trees, which is a primary reason for exploring other families of cryptographic accumulators and Vector Commitments.

Definition 1.2. An argument system between a Prover and a Verifier is *non-interactive* if it consists of a single round.

Definition 1.3. An argument of knowledge is said to be *public coin* if all challenges sent from the Verifier to the Prover are chosen uniformly at random and independently of the Prover's messages.

If the challenges are public coin, any interactive argument of knowledge can be converted into a non-interactive argument using a Fiat-Shamir heuristic ([FS86]). We now briefly introduce pairings.

Definition 1.4. For abelian groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, a *pairing*

$$\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$$

is a map with the following properties.
1. Bilinearity: $\mathbf{e}(x_1 + x_2, y_1 + y_2) = \mathbf{e}(x_1, y_2) * \mathbf{e}(x_1, y_2) * \mathbf{e}(x_2, y_1) * \mathbf{e}(x_2, y_2)$
$\forall\, x_1, x_2 \in \mathbb{G}_1,\ y_1, y_2 \in \mathbb{G}_2$.
2. Non-degeneracy: The image of $\mathbf{e}$ is non-trivial.
3. Efficient computability.

In pairing-based cryptography, we typically work in settings where the groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic of order p for some 256-bit prime p so as to have a 128-bit security level. Such pairings $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ are classified into three types:
- Type I: $\mathbb{G}_1 = \mathbb{G}_2$.
- Type II: $\mathbb{G}_1 \neq \mathbb{G}_2$ but there is an *efficiently computable* isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.
- Type III: There is no *efficiently computable* isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

In practice, the groups $\mathbb{G}_1, \mathbb{G}_2$ are cyclic subgroups of the p-torsion subgroup of some pairing-friendly elliptic curve over a prime field $\mathbb{F}_l$ and $\mathbb{G}_T$ is the group of p-th roots of unity in the algebraic closure $\overline{\mathbb{F}_l}$. The pairing is usually the (alternating) Weil pairing which is efficiently computable with Miller's algorithm. If the elliptic curve is supersingular, a symmetric pairing on the group of p-torsion points can be derived by composing the Weil pairing with an appropriate endomorphism of the elliptic curve. No such symmetric pairing exists for ordinary elliptic curves (as far as we know).

1.2 Cryptographic assumptions

We state the computationally infeasible problems that the security of our constructions depends on.

Definition 1.5. n-strong Diffie Hellman assumption: *Let $\mathbb{G}$ be a cyclic group of prime order p generated by an element g, and let $s \in \mathbb{F}_p^*$. Any probabilistic polynomial-time algorithm that is given the set $\{g^{s^i} : 1 \leq i \leq n\}$ can find a pair $(a, g^{1/(s+a)}) \in \mathbb{F}_p^* \times \mathbb{G}$ with probability at most $\mathbf{O}(\frac{1}{p})$.*

The following lemma follows immediately from the definition.

Lemma 1.1. *Let $\alpha \in \mathbb{F}_p$ and let $f(X)$ be any polynomial in $\mathbb{F}_p[X]$ not divisible by $(X + \alpha)$. Under the n-strong Diffie Hellman assumption, no probabilistic polynomial time algorithm can compute an element w such that $w^{s+\alpha} = g^{f(s)}$.*

Proof. Suppose a probabilistic polynomial time (PPT) algorithm does produce an element $w \in \mathbb{G}$ $w^{s+\alpha} = g^{f(s)}$. Since the polynomials $f(X)$, $X + \alpha$ are relatively prime, we may compute polynomials $h_1(X), h_2(X)$ such that

$$f(X)h_1(X) + (X + \alpha)h_2(X) = 1, \ \deg h_1(X) = 0.$$

Set $\widetilde{w} := w^{h_1(s)} g^{h_2(s)}$. Then $\widetilde{w}^{(s+\alpha)f(s)} = g$, which contradictions the n-strong Diffie Hellman assumption. $\square$

Definition 1.6. Knowledge of exponent assumption:. *Let $\mathbb{G}$ be a cyclic group of prime order p generated by an element g, and let $s \in \mathbb{F}_p^*$. Suppose there exists a PPT algorithm $\mathcal{A}_1$ that given the set $\{g^{s^i}, g^{s^i \alpha} : 1 \leq i \leq n\}$, outputs a pair $(c_1, c_2) \in \mathbb{G} \times \mathbb{G}$ such that $c_2 = c_1^{\alpha}$. Then there exists a PPT algorithm $\mathcal{A}_2$ that, with overwhelming probability, outputs a polynomial $f(X) \in \mathbb{F}_p[X]$ of degree $\leq n$ such that $c_1 = g^{f(s)}, c_2 = g^{\alpha f(s)}$.*

2 Bilinear accumulators

We describe the setup in this section.

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic groups of order p for some prime p such that there exists a pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ which is *bilinear*, *non-degenerate* and *efficiently computable*. Fix generators g_1, g_2 of the cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ respectively. Then $\mathbf{e}(g_1, g_2)$ is a generator of $\mathbb{G}_T$. Unlike in the case of RSA or class group accumulators, the (common) order of the groups is public. Instead, the secret trapdoor is an integer s in the range $[1, p - 1]$ which will serve as the private key. Unfortunately, the generation of the private key requires a trusted setup. This can be partially mitigated by using a secure multi-party computation ([BGM17]).

For a data set $\mathcal{D} = \{d_1, \cdots, d_n\}$, we define the accumulated digest

$$\mathrm{Acc}(\mathcal{D}) := g_1^{\prod\limits_{d\in\mathcal{D}}(d+s)} \in \mathbb{G}_1.$$

For a subset $\mathcal{D}_0 \subseteq \mathcal{D}$, the witness for $\mathcal{D}_0$ is defined by

$$\mathrm{Wit}(\mathcal{D}_0) := g_1^{\prod\limits_{d\in\mathcal{D}\setminus\mathcal{D}_0}(d+s)}.$$

The Verifier then needs to check whether

$$\mathrm{Wit}(\mathcal{D}_0)^{\prod\limits_{d_0\in\mathcal{D}_0}(d_0+s)} = \mathrm{Acc}(\mathcal{D}).$$

Because of the bilinearity of the pairing, it is equivalent and usually more efficient to test the following equation:

$$\mathbf{e}(\mathrm{Wit}(\mathcal{D}_0), g_2^{\prod\limits_{d_0\in\mathcal{D}_0}(d_0+s)}) = \mathbf{e}(\mathrm{Acc}(\mathcal{D}), g_2).$$

Since none of the parties are aware of the value of s, it is necessary to broadcast the sets $\{g_1, g_1^s, \cdots, g_1^{s^n}\}$ and $\{g_2, g_2^s, \cdots, g_2^{s^n}\}$ to the Prover.

The exponent $\prod\limits_{d\in\mathcal{D}}(d+s)$ can be interpreted as a degree n polynomial in the variable s. The coefficients of the polynomial are computed with a run time of $\mathbf{O}(n\log(n))$ using the Fast Fourier transform. Furthermore, the set $\mathbb{F}_p[X]$ of polynomials with $\mathbb{F}_p$-coefficients is a principal ideal domain whose maximal ideals are those generated by the irreducible polynomials. For a data set $\mathcal{D}$, the polynomial $f(X) = \prod\limits_{d\in\mathcal{D}}(X+d)$ is monic of degree $n = |\mathcal{D}|$. Let c_i denote the coefficient of X^i, i.e. $f(X) = \sum\limits_{i=0}^{n} c_i X^i$. The coefficients can be computed in run time $\mathbf{O}(n\log(n))$ using the Fast Fourier transform. The elements

$$g_1^{f(s)} = \prod_{i=0}^{n}(g_1^{s^i})^{c_i}, \quad g_2^{f(s)} = \prod_{i=0}^{n}(g_2^{s^i})^{c_i}$$

can then be computed by any party that possesses the elements $\{g_1^{s^i}, g_2^{s^i} : 0 \le i \le n\}$. The collision-resistance of the bilinear accumulator hinges on the n-Strong Diffie Hellman assumption.

The straightforward approach would be for the Verifier to compute $g_2^{\prod\limits_{d_0\in\mathcal{D}_0}(d_0+s)} \in \mathbb{G}_2$ in order to verify the equation

$$\mathbf{e}(\mathrm{Wit}(\mathcal{D}_0), g_2^{\prod\limits_{d_0\in\mathcal{D}_0}(d_0+s)}) = \mathbf{e}(\mathrm{Acc}(\mathcal{D}), g_2).$$

This requires computing the coefficients c_i $(0 \le i \le |\mathcal{D}_0|)$ of the polynomial $\prod\limits_{d_0\in\mathcal{D}_0}(d_0+s)$. The fastest known algorithm (the Fast Fourier transform) for this has run time complexity $\mathbf{O}(|\mathcal{D}_0|\log(|\mathcal{D}_0|))$. The Verifier then needs to perform the rather expensive exponentiations $(g_2^{s^i})^{c_i}$ that have run times $\mathbf{O}(\log(c_i))$. Furthermore, this makes it necessary for the Verifier to possess the entire set $\{g_1, g_1^s, \cdots, g_1^{s^n}\}$ which is not ideal. To address this, we provide the protocols `PoE` and `PoKE` for bilinear accumulators which achieve three goals.

1. They speed up the verification process by replacing some exponentiation operations by polynomial division in $\mathbb{F}_p[X]$ which is substantially cheaper.

2. Secondly, they shift most of the computational burden from the Verifier to the Prover. This is useful in settings where the Prover has more computational power at his disposal.

3. Finally, they reduce the in initial information necessary to be broadcasted to the Verifier to the set $\{g_1, g_1^s, g_2, g_2^s\}$ of size four. This is potentially useful in networks where the the Provers are substantially fewer in number than the Verifiers.

Protocol 2.1. *Protocol* `PoE` *(proof of exponent) for pairings (interactive version):*

Parameters : A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ of groups of prime order p; generators g_1, g_2 of $\mathbb{G}_1$, $\mathbb{G}_2$ respectively; a secret element $s \in \mathbb{F}_p^*$ such that the Prover possesses the elements $\{g_1^{s^i}, g_2^{s^i} : 0 \le i \le n\}$
Inputs: $a, b \in \mathbb{G}_1$; a polynomial $f(X) \in \mathbb{F}_p[X]$ of degree $\le n$
Claim: $a^{f(s)} = b$

1. The Verifier sends an element α (the challenge) to the Prover.
2. The Prover computes a $h(X)$ and a element β such that $f(X) = (X+\alpha)h(X)+\beta$. The Prover sends $Q := a^{h(s)}$ to the Verifier.
3. The Verifier computes $\beta := f(X) \pmod{(X + \alpha)}$ and accepts if and only if the equation

$$\mathbf{e}(Q, g_2^{s+\alpha}) * \mathbf{e}(a, g_2^{\beta}) = \mathbf{e}(b, g_2)$$

holds. □

Note that because of the bilinearity of the pairing, we have

$$Q_1^{s+\alpha} a^{\beta} = b \Longleftrightarrow \mathbf{e}(Q, g_2^{s+\alpha}) * \mathbf{e}(a, g_2^{\beta}) = \mathbf{e}(b, g_2).$$

The proof consists of a single element of $\mathbb{G}_1$ and in particular, is of constant size. The protocol can be made non-interactive using the Fiat-Shamir heuristic. Instead of the Verifier producing the challenge, it is generated by a secure hashing algorithm H that outputs uniformly random elements of $\mathbb{F}_p^*$. This hashing algorithm needs to be mutually agreed upon prior to the interaction. Although the asymptotic complexity remains unchanged, this protocol swaps a few exponentiation operations in the group $\mathbb{G}_1$ with polynomial division operations which are substantially faster.

Proposition 2.2. *The protocol* `PoE` *for bilinear accumulators is secure under the n-strong Diffie Hellman assumption.*

Proof. Suppose, by way of contradiction, that a probabilistic polynomial time adversary $\mathcal{A}$ produces a fake witness (Q_1, Q_2). Now,

$$Q_2 = Q_1^s \Longleftrightarrow \mathbf{e}(Q_1, g_2^s) = \mathbf{e}(Q_2, g_2)$$

and the second equality is easily verified by the Verifier. Hence, $\mathcal{A}$ must produce an element $Q_1 \in \mathbb{G}_1$ such that $Q_1^{s+\alpha} = ba^{-\beta}$. Since the element β is uniquely determined by the element α which, in turn, is randomly generated, the probability of $\mathcal{A}$ being able to extract the $(s+\alpha)$-th root of $ba^{-\beta}$ is negligible under the n-strong Diffie Hellman assumption. □

We use the last protocol to modify the proof of membership for a data set.

Protocol 2.3. *Protocol for the membership of a set.*

Parameters : A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ of groups of prime order p; generators g_1, g_2 of $\mathbb{G}_1$, $\mathbb{G}_2$ respectively; a secret element $s \in \mathbb{F}_p^*$ such that the Prover possesses the elements $\{g_1^{s^i}, g_2^{s^i} : 0 \le i \le n\}$
Inputs: The accumulated set $\mathcal{D}$; a data set $\mathcal{D}_0$.
Claim: $\mathcal{D}_0 \subseteq \mathcal{D}$.

1. The Prover computes the polynomial $f_0(X) := \prod_{d_0 \in \mathcal{D}_0} (X + d_0)$.

2. The Prover computes

$$\mathrm{Wit}(\mathcal{D}_0) := g_1^{\prod_{d \in \mathcal{D} \setminus \mathcal{D}_0} (d+s)}.$$

3. The Prover sends the Verifier a non-interactive `PoE` for $\mathrm{Wit}(\mathcal{D}_0)^{f_0(s)} = \mathrm{Acc}(\mathcal{D})$.

4. The Verifier computes $f_0(X)$ and accepts if and only if the `PoE` withstands this scrutiny. □

Thus, the proof of membership can be verified by a Verifier who possesses the set $\{g_1, g_1^s, g_2, g_2^s\}$. We next show how the last protocol can be adapted to provide an argument of knowledge of the logarithm. The goal is to construct a protocol with communication complexity much lower than simply sending the logarithm to the Verifier. We will need this protocol to batch non-membership proofs and subsequently, to construct our Vector Commitment.

Protocol 2.4. *Protocol* `PoKE` *(proof of knowledge of the exponent) for bilinear accumulators (interactive version):*

Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover possesses a polynomial $f(X) \in \mathbb{F}_p[X]$ such that $a^{f(s)} = b$

1. The Verifier sends a challenge $\alpha \in \mathbb{F}_p^*$ to the Prover.

2. The Prover computes the polynomial $h(X) \in \mathbb{F}_p[X]$ and the element $\beta \in \mathbb{F}_p$ such that $f(X) = (X + \alpha)h(X) + \beta$. The Prover then computes $Q := a^{h(s)}$ and sends Q, β to the Verifier.

3. The Prover computes $\widetilde{g}_2 := g_2^{f(s)}$ and sends it to the Verifier.

4. The Verifier then verifies the two equations

$$\mathbf{e}(a, \widetilde{g}_2) = \mathbf{e}(b, g_2); \;\; \mathbf{e}(Q, g_2^{s+\alpha}) * \mathbf{e}(a^\beta, g_2) = \mathbf{e}(b, g_2).$$

He then accepts if and only if both equations hold. □

Note that by the bilinearity of the pairing, we have

$$\mathbf{e}(a, \widetilde{g}_2) = \mathbf{e}(b, g_2) \Longleftrightarrow \log_a(b) = \log_{g_2}(\widetilde{g}_2),$$

$$\mathbf{e}(Q, g_2^{s+\alpha}) * \mathbf{e}(a^\beta, g_2) = \mathbf{e}(b, g_2) \Longleftrightarrow Q^{s+\alpha} a^\beta = b.$$

Since the Verifier only uses public randomness for the proof of exponentiation (`PoE`) and proof of knowledge of the exponent (`PoKE`), both protocols can be made non-interactive using the Fiat-Shamir heuristic.

We now describe two attacks to show that the Protocol `PoKE` needs both of the steps 2 and 3.

Attack 1: Suppose the pairing $\mathbf{e}$ is a type I pairing, i.e. $\mathbb{G}_1 = \mathbb{G}_2$. Suppose a malicious Prover possesses a polynomial $h(X) \in \mathbb{F}_p[X]$ such that $a^{h(s)} = g_2$. He computes $Q' := b^{h(s)}$ and sends it to the Verifier. The Verifier then sees that $\mathbf{e}(a, Q') = \mathbf{e}(b, g_2)$ and is tricked into believing the veracity of the Prover's claim. Thus, this protocol is not secure for type I pairings. Clearly, a similar attack can be carried out for type II pairings as well.

Attack 2: Suppose a Prover possesses polynomials $h_1(X), h_2(X)$ such that $g_1^{h_1(s)} = a$, $g_1^{h_2(s)} = b$ and $h_1(X)$ does not divide $h_2(X)$. With overwhelming probability, the challenge $\alpha \in \mathbb{F}_p^*$ is such that the polynomials $X + \alpha$ and $h_1(X)$ are relatively prime. On receiving the challenge α, he could simply compute a polynomial $q(X) \in \mathbb{F}_p$ and an element $\beta \in \mathbb{F}_p$ such that

$$h_1(X)\beta + (X + \alpha)q(X) = h_2(X)$$

and send $Q := a^{q(s)}$, β to the Verifier. The Verifier then sees that $Q^{s+\alpha}a^{\beta} = b$ and is fooled into believing that the Prover possesses a (polynomial) discrete logarithm between a and b.

Note that when $h_1(X)$ divides $h_2(X)$, this does not constitute an attack since

$$a^{\frac{h_2(s)}{h_1(s)}} = b.$$

But in the case where $h_1(X)$ does not divide $h_2(X)$, this attack shows that it is not sufficient for the Prover to send the pair $(Q, \beta) \in \mathbb{G}_1 \times \mathbb{F}_p^*$ to the Verifier.

Proposition 2.5. *The protocol* `PoKE` *for bilinear accumulators is secure under the n-strong Diffie Hellman and KEA assumptions.*

Proof. Suppose, by way of contradiction, that a malicious Prover is able to forge a fake `PoKE`. So, with notations as in the protocol, the Prover does not possess a polynomial $f(X)$ such that $a^{f(s)} = b$ but is able to produce elements $Q \in \mathbb{G}_1$, $\widetilde{g}_2 \in \mathbb{G}_2$ and $\beta \in \mathbb{F}_p$ such that $\mathbf{e}(a, \widetilde{g}_2) = \mathbf{e}(b, g_2)$ and $Q^{s+\alpha}a^{\beta} = b$.

Unless the Prover possesses a polynomial $h(X)$ and an efficiently computable isomorphism $\phi : \mathbb{G}_2 \longrightarrow \mathbb{G}_1$ such that $a^{h(s)} = g_1 = \phi(g_2)$, the validity of the equation

$$\mathbf{e}(a, \widetilde{g}_2) = \mathbf{e}(b, g_2)$$

implies the knowledge of the polynomial discrete logarithm between a and b with overwhelming probability. So we may assume without loss of generality that the Prover possesses such a polynomial $h(X)$ and isomorphism $\phi : \mathbb{G}_2 \longrightarrow \mathbb{G}_1$, i.e. the pairing $\mathbf{e}$ is of type I or II.

Now, the Prover produces an element $Q \in \mathbb{G}_1$ and $\beta \in \mathbb{F}_p$ such that $Q^{s+\alpha} = ba^{-\beta}$. By the KEA assumption, the Prover outputs a polynomial $h_1(X) \in \mathbb{F}_p[X]$ such that $Q = g_1^{h_1(s)}$, $ba^{-\beta} = g_1^{h_1(s)s}$. So

$$b = g_1^{h(s)s}a^{\beta} = a^{h(s)h_1(s)s+\beta},$$

a contradiction. □

If the pairing $\mathbf{e}$ is a type III pairing, the protocol is substantially simpler.

Protocol 2.6. *Protocol* `PoKE` *(proof of knowledge of the exponent) for type* III *bilinear accumulators:*

Parameters: A type III pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; Inputs: Elements $a, b \in \mathbb{G}_1$
Claim: The Prover possesses a polynomial $f(X) \in \mathbb{F}_p[X]$ such that $a^{f(s)} = b$

1. The Prover computes $Q := g_2^{f(s)}$ and sends it to the Verifier.
2. The Verifier accepts if and only if $\mathbf{e}(a, Q) = \mathbf{e}(b, g_2)$. □

Thus, the proof consists of a single element of $\mathbb{G}_2$ and the Verifier's task boils down to computing just one pairing.

2.1 Aggregation of membership witnesses

Let d_1, d_2 be two accumulated data elements and w_1, w_2 their witnesses with respect to the accumulator $\mathrm{Acc}(\mathcal{D})$. Thus,

$$\mathrm{Acc}(\mathcal{D}) = w_1^{s+d_1} = w_2^{s+d_2}.$$

Set $c := (d_1 - d_2)^{-1} \in \mathbb{F}_p^*$ and $w_{1,2} := w_1^{-c}w_2^{c}$. Then $w_{1,2}^{(s+d_1)(s+d_2)} = \mathrm{Acc}(\mathcal{D})$. Thus, $w_{1,2}$ is a witness for $\{d_1, d_2\}$. This technique can be generalized to larger sets as follows.

Let $\mathcal{D}_1$, $\mathcal{D}_2$ be two disjoint accumulated data sets and w_1, w_2 their witnesses with respect to the accumulator $\mathrm{Acc}(\mathcal{D})$. For brevity, we write

$$f_1(X) := \prod_{d_1 \in \mathcal{D}_1} (X + d_1), \quad f_2(X) := \prod_{d_2 \in \mathcal{D}_2} (X + d_2).$$

Thus,

$$\mathrm{Acc}(\mathcal{D}) = w_1^{f_1(s)} = w_2^{f_2(s)}.$$

Since the polynomials $f_1(X), f_2(X)$ are relatively prime and $\mathbb{F}_p[X]$ is a principal ideal domain, we may compute polynomials $h_1(X), h_2(X) \in \mathbb{F}_p[X]$ such that

$$h_1(X)f_1(X) + h_2(X)f_2(X) = 1 \in \mathbb{F}_p[X], \quad \deg h_i(X) < \deg f_{3-i}(X).$$

Furthermore, the run time complexity of computing these polynomials is $\mathbf{O}(N \log^2(N) \log(\log(N))$ where $N = \max(\deg f_1(X), \deg f_2(X))$. Now, the element $w_{1,2} := w_1^{h_2(s)} w_2^{h_1(s)}$ is such that $w_{1,2}^{f_1(s)f_2(s)} = \mathrm{Acc}(\mathcal{D})$. Thus, $w_{1,2}$ is a witness for the union $\mathcal{D}_1 \cup \mathcal{D}_2$.

2.2 Non-membership proofs

In this subsection, we show that we can have non-membership proofs of constant size with bilinear accumulators. As before, let $\mathcal{D}$ be the set of accumulated elements and let $\mathcal{D}_0$ be a set of elements disjoint from $\mathcal{D}$. For brevity, we write

$$f(X) := \prod_{d \in \mathcal{D}} (X + d), \quad f_0(X) := \prod_{d_0 \in \mathcal{D}_0} (X + d_0).$$

Since the polynomials $f(X)$, $f_0(X)$ are relatively prime, we may compute polynomials $h(X), h_1(X) \in \mathbb{F}_p[X]$ such that

$$f_0(X)h_0(X) - f(X)h(X) = 1 \in \mathbb{F}_p[X], \quad \deg h(X) < \deg f_0(X).$$

Set $w(\mathcal{D}_0) := g_1^{h_0(s)} \in \mathbb{G}_1$. Then

$$w(\mathcal{D}_0)^{f_0(s)} = \mathrm{Acc}(\mathcal{D})^{h(s)} g_1.$$

We use the pair $(w(\mathcal{D}_0), g_2^{h(s)}) \in \mathbb{G}_1 \times \mathbb{G}_2$ as the (constant-sized) non-membership witness for $\mathcal{D}_0$.

The problem that arises here is that a malicious Prover could provide a false witness since the Verifier does not know the polynomial $h(X)$. The most obvious solution to this would be to require the Prover to send $h(X)$ to the Verifier. However, that would require witnesses of size linear in the size of the set $\mathcal{D}_0$ which is not desirable. Instead, following the idea presented in [BBF19] for the `PoKE` in RSA or class group accumulators, we use the non-interactive `PoKE` for pairings to demonstrate that the element $g^{h(s)}$ was computed in an honest manner. If the pairing $\mathbf{e}$ is a type III pairing, the `PoKE` for $g^{h(s)}$ is redundant since being able to compute $g^{h(s)}$ demonstrates the knowledge of $h(X)$. We summarize the non-interactive protocol below.

Protocol 2.7. *Protocol for non-membership proofs with the bilinear accumulator*

Parameters: A pairing $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$; the accumulated data set $\mathcal{D}$; a data set $\mathcal{D}_0$ with $|\mathcal{D}_0| \leq |\mathcal{D}|$
Claim: $\mathcal{D}_0 \cap \mathcal{D} = \emptyset$.

1. The Prover computes the polynomials

$$f(X) := \prod_{d\in\mathcal{D}}(X+d), \quad f_0(X) := \prod_{d_0\in\mathcal{D}_0}(X+d_0).$$

2. The Prover computes polynomials $h(X), h_0(X) \in \mathbb{F}_p[X]$ such that

$$f_0(X)h_0(X) - f(X)h(X) = 1 \in \mathbb{F}_p[X], \quad \deg h(X) < \deg f_0(X).$$

3. The Prover computes $w_1 := g_1^{h_0(s)}$ and sends the pair

$$(w_1, w_2) := (g_1^{h_0(s)}, g_2^{h(s)}) \in \mathbb{G}_1 \times \mathbb{G}_2$$

to the Verifier.

4. The Prover provides a non-interactive `PoKE` for the equation $g_2^{h(s)} = w_2$ (redundant if $\mathbf{e}$ is a type III pairing).

5. The Prover computes $\widetilde{g}_2 := g_2^{f_0(s)}$ and sends $\widetilde{g}_2$ to the Verifier along with a non-interactive `PoE` for the equation $\widetilde{g}_2 = g_2^{f_0(s)}$.

6. The Verifier checks whether

$$w_1^{f_0(s)} = \mathrm{Acc}(\mathcal{D})^{h(s)} g_1$$

by verifying the equivalent equation

$$\mathbf{e}(w_1, \widetilde{g}_2) = \mathbf{e}(\mathrm{Acc}(\mathcal{D}), w_2) * \mathbf{e}(g_1, g_2).$$

He then verifies the non-interactive `PoKE` for $g_2^{h(s)} = w_2$ and the non-interactive `PoE` for the equation $\widetilde{g}_2 = g_2^{f_0(s)}$. He accepts if and only if all of these proofs withstand this scrutiny. □

Thus, the Verifier only needs the set $\{g_1, g_1^s, g_2, g_2^s\}$ to perform the verification. His computational burden is reduced to computing two pairings in addition to verifying a non-interactive `PoE` and a non-interactive `PoKE`. We now prove the security of this protocol under the n-strong Diffie Hellman assumption.

Theorem 2.8. *The non-membership protocol for the bilinear accumulator is secure under the n-strong Diffie Hellman and KEA assumptions.*

Proof. Recall that in Proposition 2.4, we showed that the protocol `PoKE` (and consequently, the non-interactive `PoKE`) is secure under the n-strong Diffie Hellman and KEA assumptions. Hence, if the non-interactive `PoKE` for $w_2 = g_2^{h(s)}$ is valid, the Verifier can assume that $g_2^{h(s)}$ was computed in an honest manner. Similarly, we showed that the non-interactive `PoE` is secure under the same assumptions.

Let $\mathcal{D}$ denote the accumulated data set and let $\mathcal{D}_0$ be a set such that $\mathcal{D}_0 \cap \mathcal{D}$ is non-empty. Set $f_0(X) := \prod_{d\in\mathcal{D}_0}(X+d)$. Suppose, by way of contradiction, that a probabilistic polynomial time adversary $\mathcal{A}$ produces a false non-membership witness $w(\mathcal{D}_0)$ such that $w(\mathcal{D}_0)^{f_0(s)} = \mathrm{Acc}(\mathcal{D})^{h(s)} g_1$ for some polynomial $h(X) \in \mathbb{F}_p[X]$. Choose an element $d_0 \in \mathcal{D}_0 \cap \mathcal{D}$ and write $f_0(X) = (X + d_0) f_2(X)$. Let w_0 be the membership witness of d_0, i.e.

$$w_0 := g_1^{\prod_{d\in\mathcal{D}\setminus\{d_0\}}(s+d)}.$$

Then we have

$$(w(\mathcal{D}_0)^{f_2(s)} w_0^{-h(s)})^{(s+d_0)} = g_1,$$

a contradiction since by the strong Diffie Hellman assumption, the PPT adversary outputs a $(s + d_0)$-th root of g_1 with negligible probability. □

2.3 Aggregation of non-membership witnesses

In this subsection, we show how one can aggregate witnesses for non-membership.

As before, let $\mathcal{D}$ be the accumulated data set of size n. Let $\mathcal{D}_1$, $\mathcal{D}_2$ be disjoint data sets such that $|\mathcal{D}_1| + |\mathcal{D}_1| << n$. Set

$$f_1 := \prod_{d_1 \in \mathcal{D}_1} (X + d_1), \;\; f_1 := \prod_{d_2 \in \mathcal{D}_2} (X + d_2).$$

Suppose we have non-membership witnesses $(w_1, g_2^{e_1(s)})$, $(w_2, g_2^{e_2(s)})$ for $\mathcal{D}_1$, $\mathcal{D}_2$ respectively along with non-interactive proofs of knowledge for the polynomials $e_1(X), e_2(X)$. We compute polynomials $\widetilde{f}_1(X)$, $\widetilde{f}_2(X)$ such that

$$f_1(X)\widetilde{f}_1(X) + f_2(X)\widetilde{f}_2(X) = 1, \;\; \deg \widetilde{f}_2(X) < \deg f_1(X)$$

and define

$$\widetilde{e}_{1,2}(X) := e_1(X)f_1(X)\widetilde{f}_1(X) + e_2(X)f_2(X)\widetilde{f}_2(X).$$

We compute polynomials $q(X), e_{1,2}(X) \in \mathbb{F}_p[X]$ such that

$$\widetilde{e}_{1,2}(X) = q(X)f_1(X)f_2(X) + e_{1,2}(X), \;\; \deg e_{1,2}(X) < \deg f_1(X)f_2(X).$$

Now set

$$w_{1,2} := \left(w_1^{\widetilde{f}_2(s)} w_2^{\widetilde{f}_1(s)} \right)^{q(s)}.$$

By construction,

$$w_{1,2}^{f_1(s)f_2(s)} = \mathrm{Acc}(\mathcal{D})^{e_{1,2}(s)} g_1.$$

Since we have proofs of knowledge of $e_1(X), e_2(X)$, we also have a proof of knowledge of $e_{1,2}(X)$. Thus, the pair $(w_{1,2}, g_2^{e_{1,2}(s)})$ along with a non-interactive proof of knowledge of $e_{1,2}(X)$ serves as a non-membership witness for the union $\mathcal{D}_1 \cup \mathcal{D}_2$.

2.4 Construction of the bilinear accumulator

Let E be an elliptic over a finite field $\mathbb{F}_l$ for some prime l. Let π_E be the Weil l-integer corresponding to the isogeny class of E. Then for any power l^d, the size $\#E(\mathbb{F}_{l^d})$ of the group of $\mathbb{F}_{l^d}$-rational points of E is given by

$$\#E(\mathbb{F}_{l^d}) = \left|(1 - \pi_E^d)(1 - \bar{\pi}_E^d)\right|.$$

We choose the pair (l, E) such that such that $\#E(\mathbb{F}_l)$ has a large prime divisor p and the co-factor $\frac{\#E(\mathbb{F}_l)}{p}$ is a small integer. Set $\mathbb{G}_1 := E(\mathbb{F}_l)[p]$, the p-torsion subgroup of $\#E(\mathbb{F}_l)$ and let g_1 be a generator of $\mathbb{G}_1$. Choose a point $g_2 \in E(\overline{\mathbb{F}_l})[p]$ such that g_1, g_2 are linearly independent and define $\mathbb{G}_2$ to be the group generated by g_2 so that

$$\#E(\overline{\mathbb{F}_l})[p] = \mathbb{G}_1 \times \mathbb{G}_2.$$

Define $\mathbb{G}_T$ as the group of p-th roots of unity of $\overline{\mathbb{F}_l}$. More precisely, if k is the smallest integer such that $l^k \equiv 1 \pmod p$, $\mathbb{G}_T$ is the group of p-th roots of unity of $\mathbb{F}_{l^k}$. The integer k is called the *embedding degree* of E with respect to p. The elliptic curve is said to be *pairing friendly* if the embedding degree is reasonably small for some large prime p dividing $\#E(\mathbb{F}_l)$.

Now, we have the Weil pairing

$$\mathbf{e}_{\text{Weil}} : E(\overline{\mathbb{F}_l})[p] \times E(\overline{\mathbb{F}_l})[p] \longrightarrow \mathbb{G}_T$$

which is bilinear, alternating, non-degenerate and efficiently computable by Miller's algorithm. We define the pairing

$$\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$$

as the restriction of the Weil pairing to $\mathbb{G}_1 \times \mathbb{G}_2$.

To ensure 128-bit security, the prime p needs to be of bit-size 256. The prime l needs to be substantially larger than previously thought, in light of the recent attack on the discrete logarithm problem in the target group $\mathbb{G}_T$ ([KB16]). One candidate for the elliptic curve E is the curve BLS12-381 defined over a prime field of bit-size 381 and embedding degree 12.

Symmetric pairings: Some bilinear accumulators such as the one constructed in [Ngu05] make use of symmetric bilinear pairings. To construct such a pairing, we need to choose a supersingular elliptic curve E over a field $\mathbb{F}_q$. The Weil q-integer π_E is an algebraic integer such that $\pi_E \bar{\pi}_E = q$, $[\mathbb{Q}(\pi_E) : \mathbb{Q}] = 2$ and $\pi_E^6 \in \mathbb{Z}$. We use the elliptic curve E to construct a symmetric pairing as follows.

The endomorphism ring $\text{End}(E)$ is a maximal order in the quaternion algebra $\mathbb{Q}_{l,\infty}$ ramified exclusively at l and ∞. Furthermore, $\text{End}(E)$ acts transitively on the group $E(\overline{\mathbb{F}_l})[p]$. With notations as in the preceding paragraph, we choose an endomorphism $\phi : E \longrightarrow E$ that does not preserve the cyclic group $E(\mathbb{F}_l)[p]$ and define $\mathbb{G}_2 := \phi(\mathbb{G}_1)$. We then have the symmetric pairing

$$\mathbf{e}_{\text{sym}} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T, \quad \mathbf{e}_{\text{sym}}(x, y) = \mathbf{e}(x, \phi(y)).$$

On the other hand, if E is an ordinary elliptic curve, the endomorphism ring $\text{End}(E)$ is an order in some imaginary quadratic field in which both p and l split. The group $E(\mathbb{F}_l)[p]$ is stable under the action of $\text{End}(E)$ and hence, no such symmetric pairing exists.

2.5 Complexities of operations for pairings

We briefly summarize the complexities of the operations involved in bilinear accumulators.

Let p be a prime and $f(X), g(X)$ polynomials in $\mathbb{F}_p[X]$ of degrees m, n respectively with $m \leq n$. The product $f(X)g(X)$ can be computed with $\mathbf{O}(n \log(n) \log(\log(n)))$ operations in $\mathbb{F}_p$. A Euclidean division has the same asymptotic complexity. The greatest common divisor $\gcd(f(X), g(X))$ can be computed in run time $\mathbf{O}(n \log^2(n) \log(\log(n)))$ along with polynomials $f_1(X), g_1(X)$ such that

$$\gcd(f(X), g(X)) = f_1(X)f(X) + g_1(X)g(X), \quad \deg f_1(X) \leq \deg g(X) \tag{2.1}$$

The Weil pairing is computed in run time $\mathbf{O}(\log^3(p))$ with Miller's algorithm. With Karatsuba's optimization of field multiplications, the run time can be brought down to $\mathbf{O}(\log^{2.585}(p))$.

2.6 Comparison between bilinear accumulators and RSA/class group accumulators

For 128-bit security, the bilinear accumulator requires a pairing-friendly elliptic curve over a 256-bit prime field. On the other hand, the RSA accumulator requires a 3072-bit RSA modulus and a class group accumulator requires an imaginary quadratic field with a 1600-bit discriminant. As a result, the bilinear accumulator is substantially faster when it comes to accumulation, witness generation and verification. We refer the reader to Tremel's thesis ([Tre13]) for a much more detailed comparison.

The major downside to bilinear accumulators is that the public key is linear in the size of the accumulator. This is somewhat offset by the fact that the run time of hashing data to primes for the RSA or class group accumulator is linear in the size of the accumulator. In fact, if the Prover and the Verifier perform the hashing to the primes separately, this hashing consumes more time than does the public key generation for a bilinear accumulator. Furthermore, since the primality checks are probabilistic rather than deterministic in nature, there is a small probability of hashing data to pseudo-primes rather than primes.

Unfortunately, the bilinear accumulator requires a trusted setup for the generation of the secret key. While this is far from ideal, this shortcoming can be mitigated by a secure multi-party computation which is more efficient that the one needed for the RSA key generation. Furthermore, accumulation with a *distributed* private key is more plausible with a bilinear accumulator than with a RSA accumulator.

The only known algebraic accumulator with a trustless setup is the class group accumulator which hinges on the infeasiblity of computing the class group of an imaginary quadratic field. However, the group operations in class groups of imaginary quadratic fields are too slow at the moment for deployment in the near future.

3 Vector Commitments

The aim of this section is to construct a Vector Commitment with constant sized openings using the universal bilinear accumulator constructed in the preceding section. Informally, a Vector Commitment is a binding commitment to a vector in the same way that an accumulator is a binding commitment to a set.

The first Vector Commitment with public parameters as well as openings of constant size was constructed in [BBF19] using their universal accumulator. Unfortunately, this does not seem feasible for a bilinear Vector Commitment since the bilinear accumulator has linear public parameters. But our construction does yield a bilinear VC with linear public parameters and openings of constant size which we expect to have a significant speed advantage over a RSA or class group VC. Furthermore, rather than storing the entire public parameter, the Verifier only needs to store the set $\{g_1, g_1^s, g_2, g_2^s\}$ in order to verify membership or non-membership proofs for data sets. Thus, his total amount of necessary storage is of constant size.

Definition 3.1. A *Vector Commitment* (VC) is a tuple consisting of the following PPT algorithms:

1. $\texttt{VC.Setup}(\lambda, n, \mathcal{M})$: Given security parameter λ, length n of the vector and message space $\mathcal{M}$ of vector components, output public parameters $\texttt{pp}$ which are implicit inputs to all the following algorithms.

2. $\texttt{VC.Com}(\mathbf{m}) \longrightarrow \tau$: Given an input $\mathbf{m} = (m_1, \cdots, m_n)$ output a commitment com.

3. $\texttt{VC.Update}(com, m, i, \tau)$: Given an input message m and a position i, output a commitment com and advice τ.

4. $\texttt{VC.Open}(com, m, i, \tau)$: On input $m \in \mathcal{M}$ and $i \in [1, n]$, the commitment com and advice τ, output an opening π that proves m is the i-th committed element of com.

5. $\texttt{VC.Verify}(com, m, i, \tau) \longrightarrow 0/1$: On input commitment com, an index $i \in [1, n]$ and an opening proof π, output 1 (accept) or 0 (reject).

A vector commitment is said to be a *subvector commitment* (SVC) if given a vector $\mathbf{m}$ and a subvector $\mathbf{m}'$, the committer may open the commitments at all the positions of $\mathbf{m}'$ simultaneously.

This notion was first introduced in [LM18]. It is necessary for each opening to be of size independent of the length of $\mathbf{m}'$, since otherwise it would be no more efficient than opening the positions separately. For instance, a Merkle tree is an example of a Vector Commitment that is not a subvector commitment since its position openings are not constant sized and secondly, the openings of several positions cannot be compressed into a single proof. In the rest of this section, we construct a SVC using the accumulator constructed in Section 2.

We start by constructing a bilinear accumulator as in the last section. The message space $\mathcal{M}$ is the set $\{0,1\}^*$. Our construction associates the element $i + p\mathbb{Z} \in \mathbb{F}_p^*$ for each index i of the vector. We now define a bit-vector $\mathbf{m} = (m_1, \cdots, m_{p-1})$ of length $p-1$ as follows. For each index i, we set

$$m_j = \begin{cases} 1 & \text{if } j + p\mathbb{Z} \text{ was accumulated .} \\ 0 & \text{otherwise.} \end{cases}$$

The bit-vector $\mathbf{m}$ is *sparse*, i.e. most of its entries are 0. The opening of the i-th index is a membership proof of $i + p\mathbb{Z}$ if $m_i = 1$ and a non-membership proof if $m_i = 0$. With the accumulator we constructed in the last section, each opening is of constant size. Furthermore, the openings of multiple indices can be batched into a constant sized proof by aggregating all the membership witnesses for $\mathbb{F}_p^*$-elements on the indices opened to 1 and batching all the non-membership witnesses for elements at the indices opened to 0.

We use our accumulator to commit to the set of elements corresponding to indices such that $m_i = 1$. The opening of the i-th index to m_i is an inclusion proof for d_i and the opening to $m_i = 0$ is an exclusion proof for d_i. With our bilinear accumulator, the opening of each index is constant-sized. Furthermore, the openings of multiple indices can be batched into a single constant sized proof using membership proofs for elements on the indices opened to elements of $\mathbb{F}_p^*$ and non-membership proofs for elements opened to 0.

A key-value map commitment: Following the ideas in [BBF19], we use our VC to build a key-value map as follows. The key-space is represented by the positions in the parse vector. The associated value is the data at the key's position. The complexity of the commitment is proportional to the number of bit indices set to 1 and hence, is independent of the length of the vector.

References

[BBF19] D. Boneh, B. Bunz, B. Fisch, *Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains.* Technical report, Cryptology ePrint Archive, Report 2018/1188, Tech. Rep, 2018.

[BGG17] S. Bowe, A. Gabizon, M. Green, *A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK*

[BGM17] S. Bowe, A. Gabizon, I. Myers, *Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model*

[CF13] D. Catalano, D. Fiore, *Vector commitments and their applications.* In Kaoru Kurosawa and Goichiro Hanaoka, editors, PKC 2013, volume 7778 of LNCS, pages 55–72. Springer, Heidelberg, Febru ary / March 2013.

[DT08] I. Damgard, N. Triandopolous, *Supporting Non-membership Proofs with Bilinear-map Accumulators*, Cryptology ePrint Archive: Report 2008/538 (2008)

[CPZ18] A. Chepurnoy, C. Papamanthou, Y. Zhang, *EDRAX : A Cryptocurrency with Stateless Transaction Validation*, Cryptology ePrint Archive, Report 2018/968, 2018. https: //eprint.iacr.org/2018/968.

[FST06] D. Freeman, M. Scott, E. Teske, *A taxonomy of pairing-friendly elliptic curves*

[FS87] A. Fiat, A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems.* In Andrew M. Odlyzko, editor, CRYPTO'86, volume 263 of LNCS, pages 186–194. Springer, Heidelberg, August 1987.

[KB16] T. Kim, R. Barbulescu, *Extended tower number field sieve: A new complexity for the medium prime case*

[LM18] R. Lai, G. Malavolta, *Optimal succinct arguments via hidden order groups*, Cryptology ePrint Archive, Report 2018/705, 2018. https://eprint.iacr.org/2018/705.

[Mil86] V. Miller, *Short Programs for functions on Curves*

[Ngu05] L. Nguyen, *Accumulators from bilinear pairings and applications*, CT-RSA, 3376:275–292, 2005.

[Tre13] E. Tremel, *Real world performance of cryptographic accumulators*

[Wes18] B. Wesolowski, *Efficient verifiable delay functions*, Cryptology ePrint Archive, Report 2018/623, 2018. https://eprint.iacr.org/2018/623.

Steve Thakur
Axoni Research Group
New York City
NY, USA
Email: steve.thakur@axoni.com