Batching non-membership proofs with bilinear accumulators

Steve Thakur
Axoni Research Group

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 challenges are public coin, the protocols 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 group-based accumulators 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 (preferably independent of the length of the vector). Subvector commitments [LM18] are VCs where 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, substantial progress was made in the paper [BBF19] where the authors showed how to provide 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.

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 burden from
the Verifier to the Prover. In particular, we provide a protocol to reduce the Verifier’s task of verifying membership proofs to a constant run time 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. $\mathbb{F}_q^\times$ denotes the cyclic multiplicative group of the non-zero elements of $\mathbb{F}_q$. For polynomials $f(X), g(X) \in \mathbb{F}_q[X]$, we denote by $\text{gcd}(f(X), g(X))$ the unique monic polynomial 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 of interaction.

**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 $G_1, G_2, G_T$, a pairings

$$e : G_1 \times G_2 \rightarrow G_T$$

is a map with the following properties.

1. Bilinearity: $e(x_1 + x_2, y_1 + y_2) = e(x_1, y_2) \ast e(x_1, y_2) \ast e(x_2, y_1) \ast e(x_2, y_2)$
   \hspace{1cm} $\forall x_1, x_2 \in G_1, \ y_1, y_2 \in G_2$.
2. Non-degeneracy: The image of $e$ is non-trivial.
3. Efficient computability.

In pairing-based cryptography, we typically work in settings where the groups $G_1, G_2, G_T$ are cyclic of order $p$ for some 256-bit prime $p$ so as to have a 128-bit security level. Such pairings $e : G_1 \times G_2 \rightarrow G_T$ are classified into three types:

- Type I: $G_1 = G_2$.
- Type II: $G_1 \neq G_2$ but there is an efficiently computable isomorphism between $G_1$ and $G_2$.
- Type III: There is no efficiently computable isomorphism between $G_1$ and $G_2$.

In practice, the groups $G_1, G_2$ are cyclic subgroups of the $p$-torsion subgroup of some pairing-friendly elliptic curve over a prime field $\mathbb{F}_p$ and $G_T$ is the group of $p$-th roots of unity in the algebraic closure $\overline{\mathbb{F}}_p$. The pairing is usually the (alternating) Weil pairing which is efficiently computable.
using 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 hinge
on.

Definition 1.5. \( n \)-strong Diffie Hellman assumption: Let \( G \) 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 \( (a, g^{1/(s+a)}) \in \mathbb{F}_p^* \times G \) with at most negligible
probability.

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

\[
f(X)h_1(X) + (X + \alpha)h_2(X) = 1, \quad \text{deg } h_1(X) = 0.
\]

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

Definition 1.6. Knowledge of exponent assumption: Let \( G \) 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 \( A_1 \)
that given the set \( \{ g^{s^i}, g^{s^i\alpha} : 1 \leq i \leq n \} \), outputs a pair \( (c_1, c_2) \in G \times G \) such that \( c_2 = c_1^\alpha \).
Then there exists a PPT algorithm \( A_2 \) 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 \( G_1, G_2, G_T \) be cyclic groups of order \( p \) for some
prime \( p \) such that there exists a pairing \( e : G_1 \times G_2 \rightarrow G_T \) which is bilinear, non-degenerate
and efficiently computable. Fix generators \( g_1, g_2 \) of the cyclic groups \( G_1, G_2 \) respectively. Then
\( e(g_1, g_2) \) is a generator of \( G_T \). Unlike in the case of accumulators based on groups of unknown
order, 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 \( D = \{ d_1, \cdots, d_n \} \), we define the accumulated digest

\[
\text{Acc}(D) := g_1^{\prod_{d \in D} (d+s)} \in G_1.
\]
For a subset \( D_0 \subseteq D \), the witness for \( D_0 \) is defined by
\[
\text{Wit}(D_0) := g_1^{d_0 \in D_0 (d_0 + s)}.
\]
The Verifier then needs to check whether
\[
\text{Wit}(D_0) \prod_{d \in D} (d_0 + s) = \text{Acc}(D).
\]
Because of the bilinearity of the pairing, it is equivalent and usually more efficient to test the following equation:
\[
e(\text{Wit}(D_0), g_2^{d_0 \in D_0 (d_0 + s)}) = e(\text{Acc}(D), g_2).
\]
Since none of the parties are aware of the value of \( s \), it is necessary to broadcast the sets \( \{g_1, g_1^2, \cdots, g_1^n\} \) and \( \{g_2, g_2^2, \cdots, g_2^n\} \) to the Prover.

The exponent \( \prod_{d \in D} (d + s) \) can be interpreted as a degree \( n \) polynomial in the variable \( s \). The coefficients of the polynomial are computed with a run time of \( O(n \log(n)) \) using the Fast Fourier transform. Furthermore, the set \( F_p[X] \) of polynomials with \( F_p \)-coefficients is a principal ideal domain whose maximal ideals are those generated by the irreducible polynomials. For a data set \( D \), the polynomial \( f(X) = \prod_{d \in D} (X + d) \) is monic of degree \( n = |D| \). Let \( c_i \) denote the coefficient of \( X^i \), i.e. \( f(X) = \sum_{i=0}^{n} c_i X^i \). The coefficients can be computed in run time \( O(n \log(n)) \) using the Fast Fourier transform. The elements
\[
g_1^{f(s)} = \prod_{i=0}^{n} (g_1^i)^{c_i}, \quad g_2^{f(s)} = \prod_{i=0}^{n} (g_2^i)^{c_i}
\]
can then be computed by any party that possesses the elements \( \{g_1^i, g_2^i : 0 \leq i \leq n\} \). 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 \( g_2^{d_0 \in D_0 (d_0 + s)} \in \mathbb{G}_2 \) in order to verify the equation
\[
e(\text{Wit}(D_0), g_2^{d_0 \in D_0 (d_0 + s)}) = e(\text{Acc}(D), g_2).
\]
However, this involves computing the coefficients \( c_i \) (0 \leq i \leq |D_0|) of the polynomial \( \prod_{d_0 \in D_0} (d_0 + s) \).

The fastest known algorithm (the Fast Fourier transform) for this has run time complexity \( O(|D_0| \log(|D_0|)) \) - followed by the exponentiations \( (g_2^{s_i})^{c_i} \) that have run times \( O(\log(c_i)) \). Furthermore, this makes it necessary for the Verifier to possess the entire set \( \{g_1, g_1^2, \cdots, g_1^n\} \) 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 \( F_p[X] \) 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 \( \{g_1, g_1^2, g_2, g_2^2\} \) of size four. This is potentially useful in networks where the the Provers are substantially fewer in number than the Verifiers.
Protocol 2.1. Proof of exponent for pairings (PoE):

**Parameters**: A pairing \( e : G_1 \times G_2 \rightarrow G_T \) of groups of prime order \( p \); generators \( g_1, g_2 \) of \( G_1, G_2 \) respectively; a secret element \( s \in \mathbb{F}_p^* \) such that the Prover possesses the elements \( \{g_1^i, g_2^i : 0 \leq i \leq n\} \)

**Inputs**: \( a, b \in G_1 \); a polynomial \( f(X) \in \mathbb{F}_p[X] \) of degree \( \leq n \)

**Claim**: \( a^{f(s)} = b \)

1. The Verifier sends an element \( \alpha \) (the challenge) to the Prover.
2. The Prover computes \( h(X) \) and a element \( \beta \) such that \( f(X) = (X + \alpha)h(X) + \beta \). The Prover sends \( Q := a^{h(s)} \) to the Verifier.
3. The Verifier computes \( \beta := e(Q, g_2^{s+\alpha}) \) and accepts if and only if the equation

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

holds.

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

\[ Q_1^{s+\alpha} a^\beta = b \iff e(Q, g_2^{s+\alpha}) * e(a, g_2^\beta) = e(b, g_2) \]

The proof consists of a single element of \( G_1 \) 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 \( G_1 \) 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 \( A \) produces a fake witness \( (Q_1, Q_2) \). Now,

\[ Q_2 = Q_1 \iff e(Q_1, g_2^{s}) = e(Q_2, g_2) \]

and the second equality is easily verified by the Verifier. Hence, \( A \) must produce an element \( Q_1 \in G_1 \) such that \( Q_1^{s+\alpha} = ba^{-\beta} \). Since the element \( \beta \) is uniquely determined by the element \( \alpha \) which, in turn, is randomly generated, the probability of \( A \) 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. Membership of a set.

**Parameters**: A pairing \( e : G_1 \times G_2 \rightarrow G_T \) of groups of prime order \( p \); generators \( g_1, g_2 \) of \( G_1, G_2 \) respectively; a secret element \( s \in \mathbb{F}_p^* \) such that the Prover possesses the elements \( \{g_1^i, g_2^i : 0 \leq i \leq n\} \)

**Inputs**: A data set \( D_0 \).

**Claim**: \( D_0 \subseteq D \).

1. The Prover computes the polynomial \( f_0(X) := \prod_{d_0 \in D_0} (X + d_0) \).
2. The Prover computes
\[ \text{Wit}(D_0) := g_{d_D(D_0)}^{d+s}. \]

3. The Prover sends the Verifier a non-interactive PoE for \( \text{Wit}(D_0)^{f_0(s)} = \text{Acc}(D) \).

4. The Verifier computes \( f_0(s) \) and accepts if and only if the PoE holds up to this scrutiny.

Thus, the proof of membership can be verified by a Verifier who possesses the set \( \{g_1, g_1^*, g_2, g_2^*\} \).

We next show how the last protocol can be adapted to provide an argument of knowledge of the exponent (PoE).

### Protocol 2.4. Proof of knowledge of the exponent for bilinear accumulators (PoKE):

**Parameters:** A pairing \( E : G_1 \times G_2 \rightarrow G_T \); generators \( g_1, g_2 \) for \( G_1, G_2 \) respectively.

**Inputs:** Elements \( a, b \in G_1 \)

**Claim:** The Prover possesses a polynomial \( f(X) \in \mathbb{F}_p[X] \) such that \( a^{f(s)} = b \).

1. The Prover \( P \) computes \( \tilde{g}_2 := g_2^{f(s)} \in G_2 \) and sends it to the Verifier \( V \).
2. The Fiat-Shamir heuristic generates a challenge \( \alpha \in \mathbb{F}_p^* \).
3. \( P \) 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 \( Q, \beta \) to the \( V \).
4. \( V \) then verifies the two equations
\[ e(a, \tilde{g}_2) = e(b, g_2) \land e(Q, g_2^{s+\alpha}) \land e(a^\beta, g_2) = e(b, g_2) \]

and accepts if and only if both equations hold.

Note that by the bilinearity of the pairing, we have
\[ e(a, \tilde{g}_2) = e(b, g_2) \iff \log_a(b) = \log_{g_2}(\tilde{g}_2), \]
\[ e(Q, g_2^{s+\alpha}) \land e(a^\beta, g_2) = e(b, g_2) \iff Q^{s+\alpha}a^\beta = b. \]

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 \( E \) is a type I pairing, i.e. \( G_1 = G_2 \). 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 \( e(a, Q) = e(b, g_2) \) 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 \( g_1^{a h_1(s)} = a, g_1^{h_2(s)} = b \) 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 \( \alpha \), 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 \( Q := a^{q(s)} \), \( \beta \) 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

\[
\left. a^{h_2(s)/h_1(s)} \right. = b.
\]

But in the case that \( h_1(X) \) does not divide \( h_2(X) \), this attack shows that it is not sufficient for the Prover to send the pair \( (Q, \beta) \in G_1 \times F_p \) 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 \( Q \in G_1, \tilde{g}_2 \in G_2 \) and \( \beta \in F_p \) such that \( e(a, \tilde{g}_2) = e(b, g_2) \) and \( Q^{s+\alpha}a^\beta = b \).

Unless the Prover possesses a polynomial \( h(X) \) and an efficiently computable isomorphism \( \phi : G_2 \rightarrow G_1 \) such that \( a^{h(s)} = g_1 = \phi(g_2) \), the validity of the equation

\[
e(a, \tilde{g}_2) = e(b, g_2)
\]

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 \( \phi : G_2 \rightarrow G_1 \), i.e. the pairing \( e \) is of type I or II.

Now, the Prover produces an element \( Q \in G_1 \) and \( \beta \in F_p \) such that \( Q^{s+\alpha} = ba^{-\beta} \). By the KEA assumption, the Prover outputs a polynomial \( h_1(X) \in F_p[X] \) 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. \( \Box \)

If the pairing \( e \) 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 \( e : G_1 \times G_2 \rightarrow G_T \); generators \( g_1, g_2 \) for \( G_1, G_2 \) respectively.

**Inputs:** Elements \( a, b \in G_1 \)

**Claim:** The Prover possesses a polynomial \( f(X) \in F_p[X] \) such that \( a^{f(s)} = b \)

1. The Verifier generates a random element \( \tilde{g}_2 \in G_2 \) and sends it to the Prover.
2. The Prover computes \( Q := e_{\tilde{g}_2}^{f(s)} \) and sends it to the Verifier.
3. The Verifier accepts if and only if \( e(a, Q) = e(b, g_2) \).

Thus, the proof consists of a single element of \( G_2 \) 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 \( \text{Acc}(D) \). Thus,

\[
\text{Acc}(D) = w_1^{s+d_1} = w_2^{s+d_2}.
\]

Set \( c := (d_1 - d_2)^{-1} \in F_p^* \) and \( w_{1,2} := w_1^{-c}w_2^c \). Then \( w_{1,2}^{(s+d_1)(s+d_2)} = \text{Acc}(D) \). Thus, \( w_{1,2} \) is a witness for \( \{d_1, d_2\} \). This technique can be generalized to larger sets, but the runtime for doing so is linear in the size of the accumulated set.
Let $D_1, D_2$ be two disjoint accumulated data sets and $w_1, w_2$ their witnesses with respect to the accumulator $\text{Acc}(D)$. For brevity, we write

$$f_1(X) := \prod_{d_1 \in D_1} (X + d_1), \quad f_2(X) := \prod_{d_2 \in D_2} (X + d_2).$$

Thus,

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

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 $O(N \log^2(N) \log(\log(N)))$ 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 $w_{1,2}^{f_1(s)f_2(s)} = \text{Acc}(D)$. Thus, $w_{1,2}$ is a witness for the union $D_1 \cup D_2$.

In order to compute the elements $w_{1-i}(s)$ (where $i = 0, 1$), the Prover needs to know (or compute from scratch) the polynomials $e_i(s)$ such that $w_{1-i} = g^{e_i(s)}$ and then compute

$$w_{1-i}^{f_i(s)} = g^{e_i(s)f_i(s)}.$$ 

The runtime complexity for this is linear in the number of elements in the accumulator. This is a major drawback when compared to accumulators based on hidden order groups.

### 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 $D$ be the set of accumulated elements and let $D_0$ be a set of elements disjoint from $D$. For brevity, we write

$$f(X) := \prod_{d \in D} (X + d), \quad f_0(X) := \prod_{d_0 \in D_0} (X + d_0).$$

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 $w(D_0) := g_1^{h_0(s)} \in \mathbb{G}_1$. Then

$$w(D_0)^{f_0(s)} = \text{Acc}(D)^{h(s)}g_1.$$ 

We use the pair $(w(D_0), g_2^{h(s)}) \in \mathbb{G}_1 \times \mathbb{G}_2$ as the (constant-sized) non-membership witness for $D_0$.

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 $D_0$ which is not desirable. Instead, following the idea presented in [BBF19] for the PoKE in accumulators based on groups of unknown order, 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 $e$ 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 \( e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T \); 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. \( \mathcal{P} \) computes polynomials \( f_0(X), h(X) \in \mathbb{F}_p[X] \) 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. \( \mathcal{P} \) 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 \( \mathcal{V} \).

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

5. \( \mathcal{P} \) computes \( \tilde{g}_2 := g_2^{f_0(s)} \) and sends \( \tilde{g}_2 \) to \( \mathcal{V} \) along with a non-interactive PoE for the equation \( \tilde{g}_2 = g_2^{f_0(s)} \).

6. \( \mathcal{V} \) checks whether
   \[ w_1^{f_0(s)} = \text{Acc}(\mathcal{D})^{h(s)}g_1 \]
   by verifying the equivalent equation
   \[ e(w_1, \tilde{g}_2) = e(\text{Acc}(\mathcal{D}), w_2) \ast e(g_1, g_2). \]

He then verifies the non-interactive PoKE for \( g_2^{h(s)} = w_2 \) and the non-interactive PoE for the equation \( \tilde{g}_2 = g_2^{f_0(s)} \). He accepts if and only if all of these proofs hold up to this scrutiny. \( \square \)

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 \( \mathcal{D}_0 \cap \mathcal{D} = \text{non-empty} \). 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 \( w(\mathcal{D}_0)^{f_0(s)} = \text{Acc}(\mathcal{D})^{h(s)}g_1 \).
for some polynomial $h(x) \in \mathbb{F}_p[x]$. Choose an element $d_0 \in D_0 \cap D$ and write
$f_0(x) = (x + d_0) f_2(x)$. Let $w_0$ be the membership witness of $d_0$, i.e.

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

Then we have

$$(w(D_0) f_2(s) w_0^{-h(s)} (s+d_0)) = g_1,$$

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.

\[ \square \]

### 2.3 Aggregation of non-membership witnesses

As was the case with membership proofs, the runtime complexity for aggregating non-membership proofs is linear in the number of elements in the accumulator. As before, let $D$ be the accumulated data set of size $n$. Let $D_1, D_2$ be disjoint data sets such that $|D_1| + |D_2| < n$. Set

$$f_1 := \prod_{d \in D_1} (X + d_1), \quad f_1 := \prod_{d \in D_2} (X + d_2).$$

Suppose we have non-membership witnesses $(w_1, g_2^{e_1(s)})$, $(w_2, g_2^{e_2(s)})$ for $D_1, D_2$ respectively along with non-interactive proofs of knowledge for the polynomials $e_1(X), e_2(X)$. We compute polynomials $\tilde{f}_1(X), \tilde{f}_2(X)$ such that

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

and define

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

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

$$e_{1,2}(X) = q(X) f_1(X) f_2(X) + e_{1,2}(X), \quad \deg e_{1,2}(X) < \deg f_1(X) f_2(X).$$

Now set

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

By construction,

$$w_{1,2}^{e_{1,2}(s)} = \text{Acc}(D)^{e_{1,2}(s)} g_1.$$ 

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 $D_1 \cup D_2$.

### 2.4 Construction of the bilinear accumulator

Let $E$ be an elliptic over a finite field $\mathbb{F}_l$ for some prime $l$. Let $\pi_E$ 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

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

10
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 \(G_1 := E(\mathbb{F}_l)[p]\), the \(p\)-torsion subgroup of \(#E(\mathbb{F}_l)\) and let \(g_1\) be a generator of \(G_1\). Choose a point \(g_2 \in E(\mathbb{F}_l)[p]\) such that \(g_1, g_2\) are linearly independent and define \(G_2\) to be the group generated by \(g_2\) so that

\[
#E(\mathbb{F}_l)[p] = G_1 \times G_2.
\]

Define \(G_T\) as the group of \(p\)-th roots of unity of \(\mathbb{F}_l\). More precisely, if \(k\) is the smallest integer such that \(l^k \equiv 1 \pmod{p}\), \(G_T\) is the group of \(p\)-th roots of unity of \(\mathbb{F}_p\). 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 suitably large for some prime \(p\) dividing \(#E(\mathbb{F}_l)\).

Now, we have the Weil pairing

\[
e_{\text{Weil}} : E(\mathbb{F}_l)[p] \times E(\mathbb{F}_l)[p] \rightarrow G_T
\]

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

\[
e : G_1 \times G_2 \rightarrow G_T
\]

as the restriction of the Weil pairing to \(G_1 \times G_2\).

In light of recent attacks against the discrete logarithm problem in the target group \(G_T\), the field of definition \(\mathbb{F}_l\) has to be larger than what was believed in the past. Instead of a 256-bit sized finite field, we would need a prime field of bit size larger than 380 to ensure 128-bit security. One candidate is the BLS12-381 which is an elliptic curve over a prime field of bit size 381 and with 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 \(\pi_E\) is an algebraic integer such that \(\pi_E \pi_E^q = q\). \([\mathbb{Q}(\pi_E) : \mathbb{Q}] = 2\) 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 \(\infty\). Furthermore, \(\text{End}(E)\) acts transitively on the group \(E(\mathbb{F}_l)[p]\). With notations as in the preceding paragraph, we choose an endomorphism \(\phi : E \rightarrow E\) that does not preserve the cyclic group \(E(\mathbb{F}_l)[p]\) and define \(G_2 := \phi(G_1)\). We then have the symmetric pairing

\[
e_{\text{sym}} : G_1 \times G_1 \rightarrow G_T, \quad e_{\text{sym}}(x, y) = e(x, \phi(y)).
\]

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 \(O(n \log(n) \log(\log(n)))\) operations in \(\mathbb{F}_p\). A Euclidean division has the same asymptotic complexity. The greatest common divisor \(\text{gcd}(f(X), g(X))\) can be computed in run time \(O(n \log^2(n) \log(\log(n)))\) along with polynomials \(f_1(X), g_1(X)\) such that

\[
\text{gcd}(f(X), g(X)) = f_1(X)f(X) + g_1(X)g(X), \quad \deg f_1(X) \leq \deg g(X)
\]
The Weil pairing is computed in run time $O(\log^3(p))$ with Miller’s algorithm. With Karatsuba’s optimization of field multiplications, the run time can be brought down to $O(\log^{2.585}(p))$.

3 Vector Commitments

The aim of this section is to construct a Vector Commitment with constant sized openings using the 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 based on groups of unknown order. 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 group-based VC. Furthermore, rather than storing the entire public parameter, the Verifier only needs to store the set $\{g_1, g_1^2, g_2, g_2^2\}$ in addition to the membership proofs which are of constant size. Thus, his total amount of storage is of constant size.

Definition 3.1. A Vector Commitment (VC) is a tuple consisting of the following PPT algorithms:
1. $\text{VC.Setup}(\lambda, n, \mathcal{M})$: Given security parameter $\lambda$, length $n$ of the vector and message space $\mathcal{M}$ of vector components, output public parameters $\text{pp}$ which are implicit inputs to all the following algorithms.
2. $\text{VC.Com}(m) \rightarrow \tau$: Given an input $m = (m_1, \ldots, m_n)$ output a commitment $\text{com}$.
3. $\text{VC.Update}(\text{com}, m, i, \tau)$: Given an input message $m$ and a position $i$, output a commitment $\text{com}$ and advice $\tau$.
4. $\text{VC.Open}(\text{com}, m, i, \tau)$: On input $m \in \mathcal{M}$ and $i \in [1, n]$, the commitment $\text{com}$ and advice $\tau$, output an opening $\pi$ that proves $m$ is the $i$-th committed element of $\text{com}$.
5. $\text{VC.Verify}(\text{com}, m, i, \tau) \rightarrow 0/1$: On input commitment $\text{com}$, an index $i \in [1, n]$ and an opening proof $\pi$, output 1 (accept) or 0 (reject).

A vector commitment is said to be a subvector commitment (SVC) if given a vector $m$ and a subvector $m'$, the committer may open the commitments at all the positions of $m'$ simultaneously. This notion was first introduced in [LM18]. It is necessary for each opening to be of size independent of the length of $m'$, 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 $m = (m_1, \ldots, m_{p-1})$ 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 $m$ 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 $F_p^*$-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-size. 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 $F_p^*$ and non-membership proofs for elements opened to 0.

3.1 A key-value map commitment

Following the ideas of [BBF19], we use our sparse VC to construct a key-value map commitment as follows. The key-space is represented by positions in the vector and the associated value is the data at the keys position. The vector length is exponential in the key length and most positions are zero. The complexity of the commitment is proportional to the number of bit indices that are set to 1 and hence, is independent of the length of the vector.

References


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


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


[Mil05] V. Miller, Short Programs for functions on Curves

[Ng00] L. Nguyen, Accumulators from bilinear pairings and applications, CTRSA, 3376:275–292, 2005

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


Steve Thakur
Axoni Research Group
New York City
NY, USA
Email: steventhakur01@gmail.com