# Dynamic Universal Accumulator with Batch Update over Bilinear Groups [⋆]

Giuseppe Vitto[1], Alex Biryukov[1]

DCS&SnT, University of Luxembourg, Esch-sur-Alzette, Luxembourg
name.surname@uni.lu

**Abstract.** We propose a Dynamic Universal Accumulator in the Accumulator Manager setting for bilinear groups which extends Nguyen's positive accumulator and Au et al. [18] and Damgård and Triandopoulos non-membership proof mechanism [16]. The new features include support for batch addition and deletion operations as well as a privacy-friendly decentralized batch witness update protocol, where the witness update information is the same for all users. Together with a non-interactive zero-knowledge protocol, these make the proposed scheme suitable as an efficient and scalable Anonymous Credential System, accessible even by low-resource users. We show security of the proposed protocol under the $t$-SDH assumption through a proper initialization of the accumulator and we demonstrate its practical relevance by providing and discussing an implementation realized using state-of-the-art libraries.

**Keywords:** accumulator, universal, dynamic, batch update, privacy-preserving KYC, anonymous credentials

## 1 Introduction

A cryptographic accumulator allows to aggregate many different values from a finite set into a fixed-length digest called *accumulator value*. Differently than hash functions, accumulators permit to further verify if an element is either accumulated or not in a given accumulator value by using the so-called *membership* and *non-membership witnesses*, respectively. Accumulator schemes which support membership witnesses are referred to as *positive* accumulators, the ones that support non-membership witnesses are called *negative*, while the ones that support both are called *universal* accumulators. A common requirement for the accumulator schemes is the ability to change the set of accumulated elements, hence permitting *accumulator updates*: when the accumulator allows to dynamically *add* and *delete* elements, it is said to be a *dynamic accumulator*.

Whenever addition or deletion operations occur for one or several elements (in the latter case these are called *batch additions and deletions*), already issued witnesses should be updated to be consistent with the new accumulator value.

[⋆] This work is supported by the Luxembourg National Research Fund (FNR) project FinCrypt (C17/IS/11684537).

Ideally this should be done using a short amount of *witness update data* (i.e. whose cost/size is not dependent on the number of elements involved) and with only publicly available information (i.e. without knowledge of any *secret* accumulator parameters). While there are many constructions that satisfy the public update condition, as regards to the update cost, Camacho and Hevia showed in [20] an impossibility result to have *batch witness updates* whose update data size is independent from the number of elements involved. More precisely, they showed that for an accumulator state which accumulates $n$ elements, the witness update data size for a batch delete operation involving $m$ elements cannot be less than $\Omega(m \log \frac{n}{m})$, thus requiring at least $\Omega(m)$ operations to update.

**Our Contributions.** In this paper we propose a Dynamic Universal Accumulator in the Accumulator Manager setting for pairing-friendly elliptic curves, which supports batch operations and public batch witness updates as well as privacy preserving zero-knowledge proof of knowledge for membership and non-membership witnesses. Its features are manifold:

- **Efficiency:** starting from Nguyen's positive accumulator and Au et al. [18] and Damgård and Triandopoulos non-membership proof mechanism [16], we state a Dynamic Universal Accumulator in the *Accumulator Manager* setting (i.e. it is managed by a central authority who knows the accumulator trapdoor), by using efficient and secure Type-III pairing-friendly elliptic curves.
- **Support for Batch Operations:** the underlying accumulator is then extended to fully support batch addition and deletion operations, as well as membership and non-membership batch witness updates.
- **Decentralized Batch Witness Update Protocol:** we designed a decentralized batch witness update protocol where the batch witness update information published by the Accumulator Manager after a batch operation is the same for all users. This information can be securely (and privately) pre-processed by third-party servers in order to allow users to update their witnesses in a constant number of elementary operations, even in the case many batch operations occurred from their last update. This allows the accumulator to be used even when only limited-resource devices (ex. smartphones) are available to users.
- **Optimal Batch Update:** the number of operations needed to batch update witnesses equals the lower bound given by Camacho and Hevia [20] in the case of a batch deletion operation. The same complexity holds in the case of either a batch addition operation and a batch addition & deletion operation, where $m$ new elements are added and other $m$ elements are deleted, namely $\mathcal{O}(m)$ update time for a batch witness update information size of $m \log pq$ bits, where $p$ is the size of the underlying bilinear group and $q$ is the size of the defining finite field.
- **Security:** the accumulator is shown to be collision resistant under the $t-$SDH assumption and also addresses the relevant recent attacks found in [30]. Additional security analysis which considers attackers that have access to the now public batch witness update information is also provided.

- **Zero-Knowledge Friendly:** zero-knowledge protocols are supported for any operation involving witnesses: we detail an efficient zero-knowledge proof of knowledge to show ownership of a valid witness and the batch witness update protocol is designed so that the results of a delegated batch witness update pre-processing is obtained without letting the third-party servers to learn anything except the publicly available data.
- **Implementation:** to show efficiency and its practical relevance, we implemented and benchmarked the proposed accumulator using state-of-the-art libraries for pairing-friendly elliptic curves.

It follows, that our accumulator is well suited to be the building block of an *Anonymous Credential System*, which originally motivated this work. In these systems only the users which were previously authorized by a central authority (the Accumulator Manager) can use the issued credentials to authenticate to the third-party verifier (ex. some financial service provider, like bank or an exchange). They do so by proving in zero-knowledge ownership of a valid membership or non-membership witnesses, depending if the accumulator is used as a white- or black-list. Furthermore, doing so anonymously and unlinkably, even if the verifier colludes with the accumulator manager. This could be crucial in many applications given current societal challenges of protecting user privacy on the one hand and government-imposed know-your-customer regulations on the other hand.

**Outline of the Paper.** In Section 2 we describe related work about relevant accumulator schemes, while in Section 3 we set out the notation used in the whole paper. In Section 4 we summarize Nguyen's [11] positive accumulator extended with the non-membership proof mechanism of Au et al. [18] and Damgård and Triandopoulos [16], both restated for Type-III bilinear pairings. In Section 4.1 we show *collision resistance* under the $t-$Strong Diffie-Hellman assumption of the resulting accumulator, thus showing that it is infeasible to forge membership witnesses for not-accumulated elements and, respectively, non-membership witnesses for accumulated ones. In Section 5 we further extend the accumulator scheme in order to support batch addition and deletion operations in a way that enable users to publicly update both their membership and non-membership witnesses (Section 6). Motivated by a recent cryptanalysis [30] of Au et al. accumulator scheme, we discuss in Section 7 how the proposed accumulator can be initialized in order to prevent the possibility to issue arbitrary non-membership witnesses for non-accumulated elements and thus be safe from collusion attacks. In Section 8 we fully detail a zero-knowledge protocol to show ownership of a valid witness for a given accumulator state, while in Section 9 benchmarks of a concrete implementation of the scheme are discussed. We then present our conclusions in Section 10.

## 2 Related Works

The first accumulator scheme was formalized by Benaloh and De Mare [3] in 1993 as a time-stamping protocol. Since then, many other accumulator schemes have been proposed. Currently, three main families of accumulators can be distinguished in literature: schemes designed in groups of unknown order [3, 4, 9, 13, 5, 22, 26], others designed in groups of known order [11, 16, 18, 19] and hash-based constructions [2, 6, 8, 15, 25]. Relevant to this paper are the schemes belonging to the second family, where the considered group is a prime order bilinear group.

Nguyen in [11] proposed a dynamic positive accumulator for symmetric bilinear groups, where up to $t$ elements can be accumulated assuming that the $t-$Strong Diffie-Hellman assumption holds in the underlying group. Damgård and Triandopoulos [16] extended Nguyen's scheme, under the same security assumptions, to support non-membership proofs, thus defining a *universal* accumulator based on bilinear pairings. Soon after this work, Au *et al.* [18] extended Nguyen's scheme to a universal accumulator by proposing two possible variants: the more efficient $\alpha$-based construction best suitable when a central authority $-$the Accumulator Manager$-$ keeps the accumulator updated, and the alternative more decentralized but less efficient reference string-based construction. We note that non-membership witness definition provided in the latter construction is equivalent to Damgård and Triandopoulos' one.

Recently, Biryukov, Udovenko and Vitto [30] cryptanalized both Au et al. variants and found different attacks able to either recover the accumulator secret parameter or issue arbitrary witnesses. While they consider the $\alpha$-based construction insecure, they conclude that in presence of an Accumulator Manager, it is possible to safely use the witness defining equations provided in the reference string-based construction (or equivalently, the Damgård and Triandopoulos' construction) by properly initializing the accumulator value.

The Dynamic Universal Accumulator obtained by combining Nguyen's positive accumulator and Au et al. and Damgård and Triandopoulos' non-membership witness mechanism, will be the starting point of our dynamic universal accumulator scheme, which we will further extend to support batch operations and public batch witness update.

Another approach on how to build a dynamic positive accumulator based on bilinear groups is given by Camenisch *et al.* in [19] where, alternatively to Nguyen's construction, a scheme relying on the $t-$DHE assumption is proposed.

## 3 Notation

Following the notation of [17], an efficiently computable non-degenerate bilinear map $e : G_1 \times G_2 \to G_T$ is said to be a Type-I pairing if $G_1 = G_2$, while it's called Type-III pairing if $G_1 \neq G_2$ and there are no efficiently computable isomorphisms between $G_1$ and $G_2$. We will denote with uppercase Roman letters (e.g. $P, V$) elements belonging to $G_1$ and with uppercase Roman letters with a tilde above (e.g. $\tilde{P}, \tilde{Q}$) elements in $G_2$. The identity points of $G_1$ and $G_2$ are denoted with $O$ and $\tilde{O}$, respectively.

Sets are denoted with uppercase letters in calligraphic fonts (e.g. $\mathcal{ACC}, \mathcal{Y}$) while accumulator elements are denoted with (eventually indexed) lowercase Roman letters: $y$ usually denotes the *reference element*, that is the one we take as an example to perform operations, while $y_{\mathcal{S}}$ denotes an element in the set $\mathcal{S}$. Exceptions are the membership and non-membership witness, denoted respectively with $w$ and $\bar{w}$, and the partial non-membership witness $d$.

Vectors are denoted with capital Greek letters (e.g. $\varUpsilon, \varOmega$). The vector operation $\langle \varPhi, \varPsi \rangle$ is the dot product, that is the sum of the products of the corresponding entries of $\varPhi$ and $\varPsi$, while $a \circ \varPhi$ denotes the usual scalar-vector multiplication where each entry of $\varPhi$ is multiplied by $a$.

We also use a convention that sum and the product of a sequence of terms with starting index greater than the ending one are assumed to be equal to $\sum_i^j a_i = 0$ and $\prod_i^j b_i = 1$ when $i > j$.

## 4 A Dynamic Universal Accumulator for Pairing Friendly Elliptic Curves

We now summarize Nguyen's positive accumulator scheme [11] (i.e. *Membership Witness, Update and Verification*) extended with the non-membership proof system of Au *et al.* [18] and Damgård and Triandopoulos [16] (i.e. *Non-membership Witness, Update and Verification*).

Due to recent progresses in discrete logarithm computations [23, 24, 28], which weaken the security of efficient implementable elliptic curves provided with a Type-I pairing, we restate their definitions into a Type-III setting, making it best suitable for efficient and more secure pairing-friendly elliptic curves.

In addition, we introduce new concepts (e.g. *Accumulator States, Epochs*) and parameters (e.g. `batchMax`), to make the accumulator definition coherent with the batch operations and the batch witness update protocol we will describe starting from Section 5.

**Bilinear Group Generation.**[1] Given a security parameter $1^\lambda$, generate over a prime order finite field $\mathbb{F}_q$ an elliptic curve $E(\mathbb{F}_q)$ with embedding degree $k$ which has an efficiently computable non-degenerate bilinear map $e : G_1 \times G_2 \rightarrow G_T$ such that

- $G_1$ is a subgroup of $E(\mathbb{F}_q)$.
- Letting $d$ be the cardinality of the automorphisms group of $E(\mathbb{F}_q)$, $G_2$ is a subgroup of $\tilde{E}(\mathbb{F}_{q^{k/d}})$ which is the unique degree-$d$ twist of $E$ over $\mathbb{F}_{q^{k/d}}$.
- $G_T$ is a subgroup of $(\mathbb{F}_{q^k})^*$.
- $|G_1| = |G_2| = |G_T| = p$ is prime.
- $P, \tilde{P}, e(P, \tilde{P})$ are generators of $G_1$,$G_2$,$G_T$, respectively.
- There are no efficiently computable isomorphisms between $G_1$ and $G_2$.

Then denote as $\mathbb{G} = (p, G_1, G_2, G_T, P, \tilde{P}, e)$ the resulting bilinear group.[2]

[1] We refer, for example, to [21] for more technical details on how these bilinear groups can be efficiently generated and implemented.

[2] We note that the accumulator scheme works with any bilinear group.

**Accumulator Parameters.** Randomly select an $\alpha \in (\mathbb{Z}/p\mathbb{Z})^*$ and consider $\mathcal{ACC} = (\mathbb{Z}/p\mathbb{Z})^* \setminus \{-\alpha\}$ as the domain of accumulatable elements. Moreover, set a bound `batchMax` to the maximum number of batch additions and/or deletions possible in each epoch (See Section 5).

The bilinear group $\mathbb{G}$, the bound `batchMax` and the point $\tilde{Q} = \alpha\tilde{P}$ are the accumulator *public parameters* and are available to all accumulator users, while $\alpha$ is the accumulator *secret parameter* and is known only to the Accumulator Manager.

**Accumulator Initialization.** Select a set $\mathcal{Y}_{V_0} \subset \mathcal{ACC}$ and let the initial accumulator value to be equal to

$$V_0 = \left(\prod_{y \in \mathcal{Y}_{V_0}} (y+\alpha)\right) P$$

The set $\mathcal{Y}_{V_0}$ is kept secret and its elements are never removed from the accumulator.[3]

**Accumulator States and Epochs.** An accumulator state is a pair $(V, \mathcal{Y}_V)$ where $V \in G_1$ is the corresponding accumulator value and $\mathcal{Y}_V \subseteq \mathcal{ACC}$ denotes the set of elements accumulated into $V$ (initialization elements excluded). We call *epoch* the period of time during which an accumulator state remains unchanged.

Given an accumulator state $(V, \mathcal{Y}_V)$, the accumulator value $V$ is equal to

$$V = \left(\prod_{y \in \mathcal{Y}_V} (y+\alpha)\right) V_0 = \left(\prod_{y \in \mathcal{Y}_V \cup \mathcal{Y}_{V_0}} (y+\alpha)\right) P$$

and can be computed from $\mathcal{Y}_V$ and $V_0$ only if the secret parameter $\alpha$ is known.

**Accumulator Update.** The accumulator state $(V, \mathcal{Y}_V)$ changes when one or more elements are added or removed from the accumulator. This can be done using the following single element Addition or Deletion operations.

- **Addition**: if $y \in \mathcal{ACC} \setminus \mathcal{Y}_V$, the element $y$ is added into the accumulator when the accumulator value is updated from $V$ to $V'$ as $V' = (y+\alpha)V$ It follows that $\mathcal{Y}_{V'} = \mathcal{Y}_V \cup \{y\}$.
- **Deletion**: if $y \in \mathcal{Y}_V$, the element $y$ is deleted from the accumulator when the accumulator state is updated from $V$ to $V'$ as $V' = \frac{1}{y+\alpha}V$ It follows that $\mathcal{Y}_{V'} = \mathcal{Y}_V \setminus \{y\}$.

**Membership Witness.** Let $(V, \mathcal{Y}_V)$ be an accumulator state and $y$ an element in $\mathcal{ACC}$. Then $w_{y,V}$ is a *membership witness for $y$ with respect to the accumulator value $V$* if

$$C = \frac{1}{y+\alpha}V$$

[3] The security of the scheme strongly depends on how the elements in $\mathcal{Y}_{V_0}$ are chosen. See Section 7 for a complete discussion.

and $w_{y,V} = C$.

The Accumulator Manager issues the membership witness $w_{y,V}$ to a user associated to the element $y$, in order to permit him to prove that $y$ *is* accumulated into $V$.[4]

**Membership Witness Update.** When accumulator state changes happen, users whose elements are not involved in the corresponding Addition or Deletion operations, have to update their witnesses with respect to the new accumulator state to continue being able to prove statements about their associated elements.

After an accumulator state change, users' membership witnesses are updated according to the following operations:

- **On Addition**: suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V', \mathcal{Y}_{V'})$ as a result of an Addition operation. Hence, for a certain $y' \in \mathcal{ACC} \setminus \mathcal{Y}_V$, $V' = (y' + \alpha)V$ and $\mathcal{Y}_{V'} = \mathcal{Y}_V \cup \{y'\}$.
  Then, for any $y \in \mathcal{Y}_V$, $w_{y,V} = C$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ by computing
  $$C' = (y' - y)C + V$$
  and letting $w_{y,V'} = C'$.
- **On Deletion**: suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V', \mathcal{Y}_{V'})$ as a result of a Deletion operation. Hence, for a certain $y' \in \mathcal{Y}_V$, $V' = \frac{1}{y'+\alpha}V$ and $\mathcal{Y}_{V'} = \mathcal{Y}_V \setminus \{y'\}$.
  Then, for any $y \in \mathcal{Y}_{V'}$, $w_{y,V} = C$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ computing
  $$C' = \frac{1}{y' - y}C - \frac{1}{y' - y}V'$$
  and letting $w_{y,V'} = C'$.

**Membership Witness Verification.** A membership witness $w_y = C$ for an element $y \in \mathcal{ACC}$ is *valid* for the accumulator state $(V, \mathcal{Y}_V)$ if and only if

$$e(C, y\tilde{P} + \tilde{Q}) = e(V, \tilde{P})$$

When $w_y$ is a valid membership witness for the state $(V, \mathcal{Y}_V)$ we assume that $y \in \mathcal{Y}_V$ and hence $w_y = w_{y,V}$.

**Non-Membership Witness.** Let $(V, \mathcal{Y}_V)$ be an accumulator state and $y$ an element in $\mathcal{ACC}$. Then $\bar{w}_{y,V}$ is a *non-membership witness for* $y$ *with respect to the accumulator state* $V$ if, by letting

$$f_V(x) = \prod_{y_i \in \mathcal{Y}_V \cup \mathcal{Y}_{V_0}} (y_i + x) \quad \in \quad \mathbb{Z}/p\mathbb{Z}[x]$$

[4] We note that the witness $C$ is equal to the previous accumulator state value, while $y$ can be deduced from the public witness update information. When the accumulator is employed as an authentication mechanism, single additions in place of batch operations lack users' privacy and expose to impersonation attacks.

it holds

$$d = f_V(-y) \mod p \text{ with } d \neq 0, \qquad C = \frac{f_V(\alpha) - d}{y + \alpha} P$$

and $\bar{w}_{y,V} = (C, d)$.

The Accumulator Manager issues the non-membership witness $\bar{w}_{y,V}$ to a user associated to the element $y$, in order to permit him to prove that $y$ *is not* accumulated into the accumulator value $V$.

**Non-Membership Witness Update.** After an accumulator state change, users' non-membership witnesses are updated according to the operation occurred as:

- **On Addition**: suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V', \mathcal{Y}_{V'})$ as a result of an Addition operation. Hence, for a certain $y' \in \mathcal{ACC} \setminus \mathcal{Y}_V$, $V' = (y' + \alpha)V$ and $\mathcal{Y}_{V'} = \mathcal{Y}_V \cup \{y'\}$.
  Then, for any $y \in \mathcal{Y}_V$, $\bar{w}_{y,V} = (C, d)$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ computing
  $$C' = (y' - y)C + V, \qquad d' = d \cdot (y' - y)$$
  and letting $\bar{w}_{y,V'} = (C', d')$.
- **On Deletion**: suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V', \mathcal{Y}_{V'})$ as a result of a Deletion operation. Hence, for a certain $y' \in \mathcal{Y}_V$, $V' = \frac{1}{y'+\alpha} V$ and $\mathcal{Y}_{V'} = \mathcal{Y}_V \setminus \{y'\}$.
  Then, for any $y \in \mathcal{Y}_{V'}$, $\bar{w}_{y,V} = (C, d)$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ computing
  $$C' = \frac{1}{y' - y} C - \frac{1}{y' - y} V', \qquad d' = d \cdot \frac{1}{y' - y}$$
  and letting $\bar{w}_{y,V'} = (C', d')$.

**Non-Membership Witness Verification.** A non-membership witness $\bar{w}_y = (C, d)$ for an element $y \in \mathcal{ACC}$ is *valid* for the accumulator state $(V, \mathcal{Y}_V)$ if $d \neq 0$ and

$$e(C, y\tilde{P} + \tilde{Q}) e(P, \tilde{P})^d = e(V, \tilde{P})$$

When $\bar{w}_y$ is a valid non-membership witness for the state $(V, \mathcal{Y}_V)$ we assume that $y \notin \mathcal{Y}_V$ and hence $\bar{w}_y = \bar{w}_{y,V}$.

### 4.1 Security Proof

Since the outlined Dynamic Universal Accumulator is built on top of Nguyen's positive dynamic accumulator [11] and Au et al. [18] and Damgård and Triandopoulos' non-membership proof system [16], it follows that the security proofs provided in [11, 16] can be straightforwardly generalized to show security of our scheme under the $t-$Strong Diffie-Hellman assumption. Security of accumulator schemes is usually intended as *collision resistance*: for universal accumulators, this property requires that an adversary forges with a negligible probability in the security parameter $\lambda$ a valid membership witness for a not-accumulated element and, respectively, a non-membership witness for an accumulated element. Formal definitions, as well as the security reduction to the $t$-SDH assumption restated for Type-III pairings follow.

**Definition 1. (Collision Resistance)** *The proposed Dynamic Universal Accumulator is collision resistant if, for any probabilistic polynomial time adversary $\mathcal{A}$ that has access to an oracle $\mathcal{O}$ which returns the accumulator value resulting from the accumulation of the elements of any given input subset of $(\mathbb{Z}/p\mathbb{Z})^*$, the following probabilities*

$$\mathbb{P}\left(\begin{array}{ccc} (\mathbb{G},\alpha,\tilde{Q}) \leftarrow Gen(1^{\lambda}) & , & (y,C,\mathcal{Y}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathbb{G},\tilde{Q}) \quad : \\ \mathcal{Y} \subset (\mathbb{Z}/p\mathbb{Z})^* & \wedge & V = \left(\prod_{y_i\in\mathcal{Y}}(y_i+\alpha)\right)P \quad \wedge \\ y \in (\mathbb{Z}/p\mathbb{Z})^* \setminus \mathcal{Y} & \wedge & e(C, y\tilde{P}+\tilde{Q}) = e(V,\tilde{P}) \end{array}\right)$$

$$\mathbb{P}\left(\begin{array}{ccc} (\mathbb{G},\alpha,\tilde{Q}) \leftarrow Gen(1^{\lambda}) & , & (y,C,d,\mathcal{Y}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathbb{G},\tilde{Q}) \quad : \\ \mathcal{Y} \subset (\mathbb{Z}/p\mathbb{Z})^* & \wedge & V = \left(\prod_{y_i\in\mathcal{Y}}(y_i+\alpha)\right)P \quad \wedge \\ y \in \mathcal{Y} & \wedge & d \neq 0 \qquad \wedge \\ \multicolumn{3}{c}{e(C, y\tilde{P}+\tilde{Q})e(P,\tilde{P})^d = e(V,\tilde{P})} \end{array}\right)$$

*are both negligible functions in the security parameter $\lambda$.*

**Definition 2. ($t-$Strong Diffie-Hellman Assumption [14])** *Let $\mathcal{G}$ be a probabilistic polynomial time algorithm that, given a security parameter $1^{\lambda}$, outputs a bilinear group $\mathbb{G} = (p, G_1, G_2, G_T, P, \tilde{P}, e)$. We say that the $t-$Strong Diffie-Hellman Assumption holds for $\mathcal{G}$ with respect to an $\alpha \in (\mathbb{Z}/p\mathbb{Z})^*$ if, for any probabilistic polynomial time adversary $\mathcal{A}$ and for every polynomially bounded function $t : \mathbb{Z} \to \mathbb{Z}$, the probability $\mathbb{P}\left(\mathcal{A}(P, \alpha P, \alpha^2 P, ..., \alpha^{t(\lambda)}P, \tilde{P}, \alpha\tilde{P}) = \left(y, \frac{1}{y+\alpha}P\right)\right)$ is a negligible function in $\lambda$ for any freely chosen value $y \in \mathbb{Z}/p\mathbb{Z} \setminus \{-\alpha\}$.*

In Definition 1 the adversary has access to an oracle $\mathcal{O}$ that outputs the accumulator value $V = \left(\prod_{y\in\mathcal{Y}_V}(y+\alpha)\right)P$ for any chosen input set $\mathcal{Y}_V$. Its purpose is to model the information the adversary can eventually get by looking at the published accumulator states. Since we want to show collision resistance under the $t-$SDH assumption, we should make sure that accessing the oracle $\mathcal{O}$ is not an extra requirement for our adversary. In this respect, by denoting with $t$ the maximum number of elements allowed to be accumulated simultaneously, the following Lemma shows that accessing this oracle is equivalent to the knowledge of the set $\mathcal{RS} = \{P, \alpha P, ..., \alpha^t P\}$, a set which is assumed to be known to the adversary under the $t-$SDH assumption.

**Lemma 1.** *Having access to the oracle $\mathcal{O}$ defined in Definition 1 is equivalent to the knowledge of the set $\mathcal{RS} = \{P, \alpha P, ..., \alpha^t P\}$ where $t$ is the maximum number of elements allowed to be accumulated simultaneously.*

*Proof.* $\boxed{\Rightarrow}$ Let $y$ be a generator of $(\mathbb{Z}/p\mathbb{Z})^*$. Then the polynomials $\{1, (y + x), (y+x)\cdot(y^2+x), \ldots, \prod_{i=1}^{t}(y^i+x)\}$ form a basis for the additive vector space of polynomials in $\mathbb{Z}/p\mathbb{Z}[x]$ with degree lower equal $t$ and, hence, for any given $1 \leq i \leq t$, there exists a linear combination of these polynomials that sums up to $x^i$. It follows that, iteratively calling $\mathcal{O}$ on the sets $\mathcal{Y}_i = \{y, y^2, \ldots, y^i\}$, it is possible to write a linear combination of the $V_{\mathcal{Y}_i}$ values returned which is equal to $\alpha^i P$.

$\boxed{\Leftarrow}$ Suppose the set $\mathcal{RS}$ is known. Then, for any given $\mathcal{Y}_V \subset (\mathbb{Z}/p\mathbb{Z})^*$ with $|\mathcal{Y}_V| \leq t$, using the $\mathcal{RS}$, it is possible to compute $V = \left(\prod_{y_i \in \mathcal{Y}_V}(y_i + \alpha)\right) P = \sum_{i=0}^{|\mathcal{Y}|} c_i \cdot (\alpha^i P)$. □

We now have all the required definitions to show collision resistance of the proposed accumulator scheme under the $t-$SDH assumption.

**Theorem 1.** *Let $\mathcal{G}$ be a probabilistic polynomial time algorithm that, given a security parameter $1^\lambda$, outputs a bilinear group $\mathbb{G} = (p, G_1, G_2, G_T, P, \tilde{P}, e)$ and consider an instantiation of the Dynamic Universal Accumulator outlined in Section 4 obtained using $\mathcal{G}$ for the bilinear group generation and $\alpha \in (\mathbb{Z}/p\mathbb{Z})^*$ as the secret accumulator parameter. Then, the accumulator is collision resistant if the $t-$Strong Diffie-Hellman Assumption holds for $\mathcal{G}$ with respect to $\alpha$.*

*Proof.* We note that a solution $(y, C, d)$ for the pairing equation $e(C, y\tilde{P} + \tilde{Q})e(P, \tilde{P})^d = e(V, \tilde{P})$ is also a solution for the elliptic curve points equation $(y+\alpha)C + dP = V$. We will then prove the Theorem considering this last equation only, distinguishing between membership and non-membership witnesses.

*Membership witnesses.* By contradiction, suppose there exists a probabilistic polynomial time adversary $\mathcal{A}$ that with respect to an (non-trivial) accumulator state $(V, \mathcal{Y}_V)$ outputs with a non-negligible probability a membership witness $C \in G_1$ for an element $y \in (\mathbb{Z}/p\mathbb{Z})^* \setminus \mathcal{Y}_V$. It follows that $(y+\alpha)C = V = f_V(\alpha)P$ where $f_V(x) = \prod_{y_i \in \mathcal{Y}_V}(y_i + x)$. Since $y \notin \mathcal{Y}_V$, we have that $(y + \alpha) \nmid f_V(x)$. Using the polynomial extended Euclidean algorithm, $\mathcal{A}$ computes $g(x) \in \mathbb{Z}/p\mathbb{Z}[x]$ of degree $|\mathcal{Y}_V| - 1$ and $r \in (\mathbb{Z}/p\mathbb{Z})^*$ such that $f_V(x) = g(x) \cdot (y + x) + r$. Therefore, $C = g(\alpha)P + \frac{r}{y+\alpha}P$ and using the $\mathcal{RS} = \{P, \alpha P, \alpha^2 P, ..., \alpha^{q(\lambda)}\}$, with $|\mathcal{Y}_V| \leq q(\lambda)$, can compute $g(x)P$ and hence $\frac{1}{y+\alpha}P = r^{-1}(C - g(\alpha)P)$, contradicting the $t-$SDH assumption.

*Non-membership witnesses.* Suppose there exists a probabilistic polynomial time adversary $\mathcal{A}$ that with respect to an (non-trivial) accumulator state $(V, \mathcal{Y}_V)$ outputs with a non-negligible probability a non-membership witness $(C, d) \in G_1 \times (\mathbb{Z}/p\mathbb{Z})^*$ for an element $y \in \mathcal{Y}_V$. Then $(y + \alpha)C = f_V(\alpha)P - dP$. Now, since $(y + x)|f_V(x)$ we have that $(y + x) \nmid f_V(x) - d$ for any $d \neq 0$. Thus, similarly as done before, $\mathcal{A}$ uses the polynomial extended Euclidean algorithm to compute $g(x) \in \mathbb{Z}/p\mathbb{Z}[x]$ of degree $|\mathcal{Y}_V| - 1$ and $r \in (\mathbb{Z}/p\mathbb{Z})^*$ such that

$f_V(x) - d = g(x) \cdot (y + x) + r$. Therefore, $C = g(\alpha)P + \frac{r}{y+\alpha}P$ and, using the $\mathcal{RS}$, $\mathcal{A}$ can compute $\frac{1}{y+\alpha}P = r^{-1}(C - g(\alpha)P)$, contradicting the $t-$SDH assumption. □

## 5 Batch Operations

We now describe how the Dynamic Universal Accumulator defined in previous Section can be further extended to coherently support batch addition and deletions operations both for accumulator and users' witnesses update.

We start by defining a family of polynomials which will help us show in a compact way correctness of our batch operations with respect to the underlying accumulator scheme.

**Batch Polynomials.** Given the secret accumulator parameter $\alpha$ and two disjoint sets $\mathcal{A}, \mathcal{D} \subseteq \mathbb{Z}/p\mathbb{Z}$ where $\mathcal{A} = \{y_{\mathcal{A},1}, \ldots, y_{\mathcal{A},n}\}$ and $\mathcal{D} = \{y_{\mathcal{D},1}, \ldots, y_{\mathcal{D},m}\}$, we define the following polynomials in $\mathbb{Z}/p\mathbb{Z}$:

$$v_{\mathcal{A}}(x) \doteq \sum_{s=1}^{n} \left( \prod_{i=1}^{s-1} (y_{\mathcal{A},i} + \alpha) \prod_{j=s+1}^{n} (y_{\mathcal{A},j} - x) \right)$$

$$v_{\mathcal{D}}(x) \doteq \sum_{s=1}^{m} \left( \prod_{i=1}^{s} (y_{\mathcal{D},i} + \alpha)^{-1} \prod_{j=1}^{s-1} (y_{\mathcal{D},j} - x) \right)$$

$$v_{\mathcal{A},\mathcal{D}}(x) \doteq v_{\mathcal{A}}(x) - v_{\mathcal{D}}(x) \cdot \prod_{i=1}^{n} (y_{\mathcal{A},i} + \alpha)$$

$$d_{\mathcal{A}}(x) \doteq \prod_{t=1}^{n} (y_{\mathcal{A},t} - x) \ , \qquad d_{\mathcal{D}}(x) \doteq \prod_{t=1}^{m} (y_{\mathcal{D},t} - x)$$

**Accumulator Batch Update.** Several elements are added into or removed from the accumulator using the following Batch Addition and Batch Deletion operations.

- **Batch Addition**: if $\mathcal{A} = \{y_{\mathcal{A},1}, \ldots, y_{\mathcal{A},n}\} \subseteq \mathcal{ACC} \setminus \mathcal{Y}_V$, the elements in $\mathcal{A}$ are *batch added* into the accumulator when the accumulator value is updated from $V$ to $V'$ as $V' = d_{\mathcal{A}}(-\alpha) \cdot V$. It follows that $\mathcal{Y}_{V'} = \mathcal{Y}_V \cup \mathcal{A}$.
- **Batch Deletion**: if $\mathcal{D} = \{y_{\mathcal{D},1}, \ldots, y_{\mathcal{D},m}\} \subseteq \mathcal{Y}_V$, the elements in $\mathcal{D}$ are *batch deleted* from the accumulator when the accumulator state is updated from $V$ to $V'$ as $V' = \frac{1}{d_{\mathcal{D}}(-\alpha)} \cdot V$. It follows that $\mathcal{Y}_{V'} = \mathcal{Y}_V \setminus \mathcal{D}$.
- **Batch Addition & Deletion**: if $\mathcal{A} = \{y_{\mathcal{A},1}, \ldots, y_{\mathcal{A},n}\} \subseteq \mathcal{ACC} \setminus \mathcal{Y}_V$, $\mathcal{D} = \{y_{\mathcal{D},1}, \ldots, y_{\mathcal{D},m}\} \subseteq \mathcal{Y}_V$ and $\mathcal{A} \cap \mathcal{D} = \emptyset$, the elements in $\mathcal{A}$ are batch added into the accumulator and the elements in $\mathcal{D}$ are batch deleted from the accumulator when the accumulator state is updated from $V$ to $V''$ as $V'' = \frac{d_{\mathcal{A}}(-\alpha)}{d_{\mathcal{D}}(-\alpha)} \cdot V$. It follows that $\mathcal{Y}_{V''} = \mathcal{Y}_V \cup \mathcal{A} \setminus \mathcal{D}$.

**Membership Batch Witness Update.** When a batch addition or deletion changes the accumulator state, users' membership witnesses are updated according to the following operation.

- **On Batch Addition:** suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V', \mathcal{Y}_{V'})$ as a result of an Batch Addition operation. Hence, for certain $\mathcal{A} = \{y_{\mathcal{A},1}, \ldots, y_{\mathcal{A},n}\} \subseteq \mathcal{ACC} \setminus \mathcal{Y}_V$, we have $V' = d_{\mathcal{A}}(-\alpha) \cdot V$ and $\mathcal{Y}_{V'} = \mathcal{Y}_V \cup \mathcal{A}$. Then, for any $y \in \mathcal{Y}_V$, the witness $w_{y,V} = C$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ computing

  $$C' = d_{\mathcal{A}}(y) \cdot C + v_{\mathcal{A}}(y) \cdot V$$

  and letting $w_{y,V'} = C'$.

  *Proof.* For the ease of notation, we will denote the elements $y_{\mathcal{A},i}$ with $y_i$, the accumulator value corresponding to $\left(\prod_{i=1}^{j}(y_i + \alpha)\right) V$ with $V_j$ and, for any $y \in \mathcal{Y}_V$, the intermediate membership witnesses $w_{y,V_j}$ with $C_j$.
  We prove the formula by induction on $n$, the number of batch added elements:
  $\boxed{n = 1}$: We get $C_1 = V + (y_1 - y)C$, the same formula defined for the membership witness update after a single addition operation.
  $\boxed{n - 1 \to n}$: Let $b_s = \prod_{i=1}^{s-1}(y_i + \alpha) \prod_{j=s+1}^{n}(y_j - y)$. Using the inductive hypothesis for $C_{n-1}$, we have

  $$\begin{aligned} C_n &= (y_n - y)C_{n-1} + V_{n-1} \\ &= \left(\prod_{t=1}^{n}(y_t - y)\right) C + \left(\sum_{s=1}^{n-1} b_s + \prod_{t=1}^{n-1}(y_t + \alpha)\right) V \\ &= \left(\prod_{t=1}^{n}(y_t - y)\right) C + \left(\sum_{s=1}^{n} b_s\right) V \end{aligned}$$

  as required. □

- **On Batch Deletion**: suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V', \mathcal{Y}_{V'})$ as a result of a Batch Deletion operation. Hence, for certain $\mathcal{D} = \{y_{\mathcal{D},1}, \ldots, y_{\mathcal{D},m}\} \subseteq \mathcal{Y}_V$, we have $V' = \frac{1}{d_{\mathcal{D}}(-\alpha)} V$.
  Then, for any $y \in \mathcal{Y}_{V'}$, the witness $w_{y,V} = C$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ computing

  $$C' = \frac{1}{d_{\mathcal{D}}(y)} C - \frac{v_{\mathcal{D}}(y)}{d_{\mathcal{D}}(y)} V$$

  and letting $w_{y,V'} = C'$.

  *Proof.* Similarly as before, we will denote the elements $y_{\mathcal{D},i}$ with $y_i$, the accumulator value corresponding to $\left(\prod_{i=1}^{j}(y_i + \alpha)^{-1}\right) V$ with $V_j$ and, for any $y \in \mathcal{Y}_{V'}$, the intermediate membership witnesses $w_{y,V_j}$ with $C_j$.

We prove the formula by induction on $m$, the number of batch deleted elements:
$\boxed{m = 1}$: We get $C_1 = \frac{1}{y_1 - y} C - \frac{1}{(y_1 - y)(y_1 + \alpha)} V = \frac{1}{y_1 - y}(C - V_1)$, the same formula defined for the membership witness update after a single deletion operation.
$\boxed{m - 1 \to m}$: Let $b_s = \prod_{i=1}^{s} (y_i + \alpha)^{-1} \prod_{j=1}^{s-1} (y_j - y)$. Then

$$\begin{aligned}
C_m =& \frac{1}{y_m - y}(C_{m-1} - V_m) \\
=& \frac{1}{d_{\mathcal{D}}(y)} C - \frac{1}{d_{\mathcal{D}}(y)} \cdot \left( \sum_{s=1}^{m-1} b_s \right) V - \left( (y_m - y)^{-1} \prod_{i=1}^{m} (y_i + \alpha)^{-1} \right) V \\
=& \frac{1}{d_{\mathcal{D}}(y)} C - \frac{1}{d_{\mathcal{D}}(y)} \cdot \left( \sum_{s=1}^{m} b_s \right) V
\end{aligned}$$

as required. □

- **On Batch Addition & Deletion:** suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V'', \mathcal{Y}_{V'})$ as a result of a Batch Addition & Deletion operation. Hence for certain disjoint sets $\mathcal{A} = \{y_{\mathcal{A},1}, \ldots, y_{\mathcal{A},n}\} \subseteq \mathcal{ACC} \setminus \mathcal{Y}_V$ and $\mathcal{D} = \{y_{\mathcal{D},1}, \ldots, y_{\mathcal{D},m}\} \subseteq \mathcal{Y}_V$ we have $V'' = \frac{d_{\mathcal{A}}(-\alpha)}{d_{\mathcal{D}}(-\alpha)} \cdot V$.

  Then, for any $y \in \mathcal{Y}_V$, the witness $w_{y,V} = C$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ computing

$$C' = \frac{d_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)} \cdot C + \frac{\nu_{\mathcal{A},\mathcal{D}}(y)}{d_{\mathcal{D}}(y)} \cdot V$$

  and letting $w_{y,V'} = C'$.

  *Proof.* Performing a batch addition and then a batch deletion, the membership witness $w_{y,V} = C$ for $y$ with respect to the accumulator value $V$ is iteratively updated to $\bar{w}_{y,V''} = (C'', d'')$ with respect to the updated accumulator value

$$V'' = \left( \frac{\prod_{i=1}^{n} (y_{\mathcal{A},i} + \alpha)}{\prod_{i=1}^{m} (y_{\mathcal{D},i} + \alpha)} \right) V$$

  as follows

$$C \quad \xrightarrow{\text{Add}} \quad C' = d_{\mathcal{A}}(y) C + v_{\mathcal{A}}(y) V \quad \xrightarrow{\text{Delete}}$$

$$C'' = \frac{1}{d_{\mathcal{D}}(y)} C' - \frac{v_{\mathcal{D}}(y)}{d_{\mathcal{D}}(y)} V' = \frac{d_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)} C + \left( \frac{v_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)} - \frac{v_{\mathcal{D}}(y)}{d_{\mathcal{D}}(y)} \cdot \prod_{i=1}^{n} (y_{\mathcal{A},i} + \alpha) \right) V$$

  where $V' = \prod_{i=1}^{n} (y_{\mathcal{A},i} + \alpha) \cdot V$. □

**Non-Membership Batch Witness Update.** When a batch addition or deletion changes the accumulator state, users' non-membership witnesses are updated according to the following operations. Proofs of correctness are similar to the corresponding ones described in the case of membership batch witness update and so are omitted.

- **On Batch Addition:** suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V', \mathcal{Y}_{V'})$ as a result of an Batch Addition operation. Hence, for certain $\mathcal{A} = \{y_{\mathcal{A},1}, \ldots, y_{\mathcal{A},n}\} \subseteq \mathcal{ACC} \backslash \mathcal{Y}_V$, we have $V' = d_{\mathcal{A}}(-\alpha) \cdot V$ and $\mathcal{Y}_{V'} = \mathcal{Y}_V \cup \mathcal{A}$. Then, for any $y \notin \mathcal{Y}_V$, the witness $\bar{w}_{y,V} = (C, d)$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ computing
$$C' = d_{\mathcal{A}}(y) \cdot C + v_{\mathcal{A}}(y) \cdot V \ , \qquad\qquad d' = d \cdot d_{\mathcal{A}}(y)$$
and letting $\bar{w}_{y,V'} = (C', d')$.
- **On Batch Deletion**: suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V', \mathcal{Y}_{V'})$ as a result of a Batch Deletion operation. Hence, for certain $\mathcal{D} = \{y_{\mathcal{D},1}, \ldots, y_{\mathcal{D},m}\} \subseteq \mathcal{Y}_V$, we have $V' = \frac{1}{d_{\mathcal{D}}(-\alpha)} \cdot V$ and $\mathcal{Y}_{V'} = \mathcal{Y}_V \setminus \mathcal{D}$. Then, for any $y \in \mathcal{Y}_{V'}$, the witness $\bar{w}_{y,V} = C$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ computing
$$C' = \frac{1}{d_{\mathcal{D}}(y)} \cdot C - \frac{v_{\mathcal{D}}(y)}{d_{\mathcal{D}}(y)} \cdot V \ , \qquad\qquad d' = d \cdot \frac{1}{d_{\mathcal{D}}(y)}$$
and letting $\bar{w}_{y,V'} = (C', d')$.
- **On Batch Addition & Deletion:** suppose the accumulator state changes from $(V, \mathcal{Y}_V)$ to $(V'', \mathcal{Y}_{V'})$ as a result of a Batch Addition & Deletion operation. Hence for certain disjoint sets $\mathcal{A} = \{y_{\mathcal{A},1}, \ldots, y_{\mathcal{A},n}\} \subseteq \mathcal{ACC} \setminus \mathcal{Y}_V$ and $\mathcal{D} = \{y_{\mathcal{D},1}, \ldots, y_{\mathcal{D},m}\} \subseteq \mathcal{Y}_V$ we have $V'' = \frac{d_{\mathcal{A}}(-\alpha)}{d_{\mathcal{D}}(-\alpha)} \cdot V$.

  Then, for any $y \in \mathcal{Y}_V$, the witness $\bar{w}_{y,V} = (C, d)$ is updated with respect to the accumulator state $(V', \mathcal{Y}_{V'})$ computing
$$C' = \frac{d_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)} \cdot C + \frac{\nu_{\mathcal{A},\mathcal{D}}(y)}{d_{\mathcal{D}}(y)} \cdot V \ , \qquad\qquad d' = d \cdot \frac{d_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)}$$
and letting $\bar{w}_{y,V'} = (C', d')$.

## 6 The Batch Witness Update Protocol

Users cannot batch update their witnesses directly using the formulae defined in previous section, since they would need the secret parameter $\alpha$. However, starting from their definition, the Accumulator Manager can efficiently compute and publish some update information (more precisely, the polynomials $d_{\mathcal{A}}(x), d_{\mathcal{D}}(x)$ and an elliptic curve points vector) so that users are able to update their witnesses without requiring nor leaking (see Section 6.1) any information related to $\alpha$. This will allow to define a decentralized batch membership and non-membership witness update protocol for the proposed accumulator scheme.

### 6.1 The Batch Witness Update Information

From now on, we will focus on the Batch Witness Update Addition & Deletion polynomial $v_{\mathcal{A},\mathcal{D}}(x)$ only: indeed, the polynomials $v_{\mathcal{A}}(x)$ and $v_{\mathcal{D}}(x)$ are special cases of this more general one.

We recall that our main goal is to allow users possessing a witness $(C, d)$ for an element $y$ with respect to the accumulator value $V$ to compute the quantities

$$C' = \frac{d_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)} \cdot C + \frac{\nu_{\mathcal{A},\mathcal{D}}(y)}{d_{\mathcal{D}}(y)} \cdot V \ , \qquad d' = d \cdot \frac{d_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)}$$

We note that the Accumulator Manager cannot publish all the polynomials $d_{\mathcal{A}}(x)$, $d_{\mathcal{D}}(x)$ and $v_{\mathcal{A},\mathcal{D}}(x)$, because their coefficients can leak some information related to the secret accumulator parameter $\alpha$. To give an example, suppose that after a batch addition operation, the Accumulator Manager publishes the polynomials $v_{\mathcal{A}}(x)$ and $d_{\mathcal{A}}(x)$, defined as above, with $|\mathcal{A}| > 1$. Doing simple algebra, we find that the coefficient of the $(|\mathcal{A}| - 2)-$degree monomial of $v_{\mathcal{A}}(x)$ is equal to $\alpha + \sum_{y_A \in \mathcal{A}} y_A$ : extracting the roots of $d_{\mathcal{A}}(x)$ in $\mathbb{Z}/p\mathbb{Z}$ we obtain all the elements in $\mathcal{A}$ and hence the secret parameter $\alpha$.

Leakages about $\alpha$ can be prevented by requiring the Accumulator Manager to publish in place of $v_{\mathcal{A},\mathcal{D}}(x)$, the vector of elliptic curve points

$$\Omega = \Omega_{\mathcal{A},\mathcal{D},V} = (\ c_0 V,\ c_1 V,\ \dots\ ,\ c_{\texttt{batchMax}} V\ )$$

where $v_{\mathcal{A},\mathcal{D}}(x) = \sum_{i=0}^{\texttt{batchMax}} c_i x^i$ and $c_i = 0$ if $i > max(|\mathcal{A}|, |\mathcal{D}|)$.

Users can then update their membership witness $w_{y,V} = C$ to $w_{y,V'} = C'$ by first evaluating the two polynomials $d_{\mathcal{A}}(x)$ and $d_{\mathcal{D}}(x)$ in the element $y$ and then computing

$$C' = \frac{d_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)} \cdot C + \frac{1}{d_{\mathcal{D}}(y)} \cdot \langle \Upsilon_y, \Omega \rangle$$

where $\Upsilon_y = (1, y, y^2, ..., y^{\texttt{batchMax}})$ and $\langle \cdot, \cdot \rangle$ denotes the dot product.

Similarly, a non-membership witness $\bar{w}_{y,V} = (C, d)$ is updated to $\bar{w}_{y,V'} = (C', d')$ by computing

$$C' = \frac{d_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)} \cdot C + \frac{1}{d_{\mathcal{D}}(y)} \cdot \langle \Upsilon_y, \Omega \rangle \ \ , \qquad d' = d \cdot \frac{d_{\mathcal{A}}(y)}{d_{\mathcal{D}}(y)}$$

In this scenario, assuming the Discrete Logarithm Problem to be hard in $G_1$ (a weaker assumption with respect to the $t-$SDH assumption under which accumulator collision resistance is shown), from the published $\Omega$, $d_{\mathcal{A}}(x)$ and $d_{\mathcal{D}}(x)$ it is only possible, performing roots extraction on the polynomials, to compute the respective sets $\mathcal{A}$ and $\mathcal{D}$ of batch added and batch deleted elements.

It follows that witness update operations can be performed either autonomously by users or by delegating to third-party servers the computation of (some of) the values $\langle \Upsilon_y, \Omega \rangle$, $d_{\mathcal{A}}(y), d_{\mathcal{D}}(y)$. Indeed, since the required updating values are decoupled from users' previous witnesses, *trusted* third-party servers which are

| Epoch | Accumulator State | Witness Update Information | | |
|---|---|---|---|---|
| 0 | $(V_0, \emptyset)$ | | | |
| 1 | $(V_1, \mathcal{Y}_{V_1})$ | $\Omega_1$ | $d_{\mathcal{A}_1}(x)$ | $d_{\mathcal{D}_1}(x)$ |
| $\vdots$ | $\vdots$ | | $\vdots$ | |
| $i-1$ | $(V_{i-1}, \mathcal{Y}_{V_{i-1}})$ | $\Omega_{i-1}$ | $d_{\mathcal{A}_{i-1}}(x)$ | $d_{\mathcal{D}_{i-1}}(x)$ |
| $i$ | $(V_i, \mathcal{Y}_{V_i})$ | $\Omega_i$ | $d_{\mathcal{A}_i}(x)$ | $d_{\mathcal{D}_i}(x)$ |

**Table 1.** Data published by the Accumulator Manager in each epoch.

asked to compute the witness updating values with respect to an element $y$, cannot impersonate the corresponding user, since they don't know any previous valid witness for $y$.

In the case of *untrusted* third-party servers, it is possible to use Oblivious Polynomial Evaluation techniques such as [12], [27] and [7] to delegate the computation of the elliptic curve point $\langle \Upsilon_y, \Omega \rangle = v_{\mathcal{A},\mathcal{D}}(y) \cdot V$ and of the values $d_{\mathcal{A}}(y)$ and $d_{\mathcal{D}}(y)$, in a way that third-party will not learn anything about $y$. This can be useful especially in the case of low-resource devices, thus permitting a *lightweight* privacy-preserving delegation for batch witness update operations.

### 6.2 Batch Witness Update Among Epochs

We now show how the adoption of the elliptic curve points vector $\Omega = \Omega_{\mathcal{A},\mathcal{D},V}$ not only permits the users to batch update their (non-)membership witnesses from the previous accumulator state, but also enables them to directly update from the accumulator state of any older epoch. This feature doesn't force users to permanently keep their witnesses updated to the latest accumulator state, enabling them to update their witnesses just right before they want to prove statements about the associated element $y$.

Before showing how this is possible, we extend our notation to associate accumulator and batch witness update data to a specific epoch. Given an epoch $i > 0$, we denote with $(V_i, \mathcal{Y}_{V_i})$ the corresponding accumulator state, where $\mathcal{Y}_{V_1} = \mathcal{A}_1 \setminus \mathcal{D}_1$ and $\mathcal{Y}_{V_i} = \mathcal{Y}_{V_{i-1}} \cup \mathcal{A}_i \setminus \mathcal{D}_i$ for $i > 1$, with $d_{\mathcal{A}_i}(x)$ and $d_{\mathcal{D}_i}$ the addition and deletion batch witness update polynomials, respectively, and with $\Omega_i = \Omega_{\mathcal{A}_i,\mathcal{D}_i,V_i}$. An overview of the data published by the Accumulator Manager is given in Table 1.

We further denote a membership witness $w_{y,V_i}$ for an element $y$ with respect to the accumulator value $V_i$ as $w_{y,V_i} = C_i$ and, similarly, a membership witness $\bar{w}_{y,V_i}$ as $\bar{w}_{y,V_i} = (C_i, d_i)$.

**Epoch Witnesses Batch Update.** A user who owns a valid non-membership witness $\bar{w}_{y,V_i} = (C_i, d_i)$ (resp. a valid membership witness $w_{y,V_i} = C_i$) with respect to the accumulator state $(V_i, \mathcal{Y}_{V_i})$ can update it to $\bar{w}_{y,V_j} = (C_j, d_j)$

(resp. $w_{y,V_j} = C_j$ ), for any $j > i$, as

$$C_j = \frac{d_{\mathcal{A}_{i\to j}}(y)}{d_{\mathcal{D}_{i\to j}}(y)} \cdot C_i + \frac{1}{d_{\mathcal{D}_{i\to j}}(y)} \cdot \langle\, \varUpsilon_y \;,\; \varOmega_{i\to j}(y) \,\rangle \;, \qquad d_j = d_i \cdot \frac{d_{\mathcal{A}_{i\to j}}(y)}{d_{\mathcal{D}_{i\to j}}(y)}$$

where

$$d_{\mathcal{A}_{a\to b}}(x) = \prod_{s=a+1}^{b} d_{\mathcal{A}_s}(x) \qquad d_{\mathcal{D}_{a\to b}}(x) = \prod_{s=a+1}^{b} d_{\mathcal{D}_s}(x)$$

$$\varOmega_{i\to j}(y) = \sum_{t=i+1}^{j} \big(d_{\mathcal{D}_{i\to t-1}}(y) \cdot d_{\mathcal{A}_{t\to j}}(y)\big) \circ \varOmega_t$$

*Proof.* We prove the result by induction on $j > i$.
$\boxed{j = i+1}$ A non-membership witness $\bar{w}_{y,V_i} = (C_i, d_i)$ is updated to $\bar{w}_{y,V_{i+1}} = (C_{i+1}, d_{i+1})$ as

$$C_{i+1} = \frac{d_{\mathcal{A}_{i+1}}(y)}{d_{\mathcal{D}_{i+1}}(y)} \cdot C_i + \frac{1}{d_{\mathcal{D}_{i+1}}(y)} \cdot \langle\, \varUpsilon_y \,,\; \varOmega_{i+1} \,\rangle \;, \qquad d_{i+1} = d_i \cdot \frac{d_{\mathcal{A}_{i+1}}(y)}{d_{\mathcal{D}_{i+1}}(y)}$$

obtaining the same result we get by using the formula for non-membership witnesses batch update.
$\boxed{j \Rightarrow j+1\text{:}}$ By inductive hypothesis, we assume the formula holds for $C_j$. Then

$$\begin{aligned}
C_{j+1} =& \frac{d_{\mathcal{A}_{j+1}}(y)}{d_{\mathcal{D}_{j+1}}(y)} \cdot C_j + \frac{1}{d_{\mathcal{D}_{j+1}}(y)} \cdot \langle\, \varUpsilon_y \,,\; \varOmega_{j+1} \,\rangle \\
=& \frac{d_{\mathcal{A}_{i\to j+1}}(y)}{d_{\mathcal{D}_{i\to j+1}}(y)} \cdot C_i + \frac{1}{d_{\mathcal{D}_{i\to j+1}}(y)} \cdot \Big\langle\, \varUpsilon_y \,,\; d_{\mathcal{A}_{j+1}}(y) \circ \varOmega_{i\to j} \,\Big\rangle \\
&+ \frac{1}{d_{\mathcal{D}_{i\to j+1}}(y)} \cdot \langle\, \varUpsilon_y \,,\; d_{\mathcal{D}_{i\to j}}(y) \circ \varOmega_{j+1} \,\rangle \\
=& \frac{d_{\mathcal{A}_{i\to j+1}}(y)}{d_{\mathcal{D}_{i\to j+1}}(y)} \cdot C_i + \frac{1}{d_{\mathcal{D}_{i\to j+1}}(y)} \cdot \langle\, \varUpsilon_y \,,\; \varOmega_{i\to j+1} \,\rangle
\end{aligned}$$

as required, since

$$\begin{aligned}
\varOmega_{i\to j+1} &= \sum_{t=i+1}^{j+1} \left( \prod_{h=i+1}^{t-1} d_{\mathcal{D}_h}(y) \prod_{k=t+1}^{j+1} d_{\mathcal{A}_k}(y) \right) \circ \varOmega_t \\
&= \left( \sum_{t=i+1}^{j} \left( \prod_{h=i+1}^{t-1} d_{\mathcal{D}_h}(y) \prod_{k=t+1}^{j+1} d_{\mathcal{A}_k}(y) \right) \circ \varOmega_t \right) + \left( \prod_{h=i+1}^{j} d_{\mathcal{D}_h}(y) \right) \circ \varOmega_{j+1} \\
&= d_{\mathcal{A}_{j+1}}(y) \circ \varOmega_{i\to j} + d_{\mathcal{D}_{i\to j}}(y) \circ \varOmega_{j+1}
\end{aligned}$$

The induction on $d_j$ is straightforward. □

## 7 Accumulator Initialization

Collision resistance under the $t-$SDH assumption (see Theorem 1), assures that forging non-membership witnesses for accumulated elements and, similarly, membership witnesses for non-accumulated elements, is possible only with negligible probability in the security parameter $\lambda$.

By the way, collision resistance doesn't take into account the possibility to forge non-membership witnesses for *"never authorized"* non-accumulated elements, i.e. elements for which the Accumulator Manager did not issue witnesses. This is relevant, for example, in the cases when the accumulator is used as an authentication mechanism and accumulated elements represents either whitelisted or blacklisted users which authenticate with respect to the accumulator value by showing possession of a valid membership or non-membership witness.

Forging witnesseses in the case when the Accumulator Manager should be the only authorized entity to do so is, in fact, what the *Witness Forgery Attack* outlined by Biryukov, Udovenko and Vitto in [30] does: a set of colluding users who share their non-membership witnesses can recover the (secret) *reference-string* sets $\mathcal{RS}_s = \{P, \alpha P, ..., \alpha^s P\}_{s>0}$, which enable them to compute membership and non-membership witnesses with respect to the latest accumulator value. Indeed, the knowledge of the set $\mathcal{RS} = \mathcal{RS}_t$ results to be functionally equivalent to the knowledge of $\alpha$: it is possible to either update the accumulator value (see Lemma 1) or issue valid membership and non membership witnesses (see the reference string $\mathcal{RS}$-based construction in [30, 18]).

The Witness Forgery Attack is possible as long as the number of colluding users is equal or greater to the number of elements added to initialize the accumulator value. In fact, the countermeasure proposed in [30] is to set an upper limit `NMWitnessesMax` to the total number of issuable non-membership witnesses and initialize the accumulator by adding at least $\texttt{NMWitnessesMax} + 1$ secret elements.

This will clearly prevent the reconstruction of the sets $\mathcal{RS}_s$, but in our protocol the attackers have access in each epoch to the witness update information (see Table 1), which in principle could help circumventing the fact that they will not be able to collect and share enough non-membership witnesses.

We will show that this is indeed possible but we will prove that it can be easily prevented by carefully choosing few of the $\texttt{NMWitnessesMax} + 1$ elements added to prevent the Witness Forgery Attack.

We start by introducing some theoretical result. The purpose of the following Proposition is to show some properties on elements that have particular multiplicative orders in the group $(\mathbb{Z}/p\mathbb{Z})^*$. These properties will be useful to prove the subsequent Theorem 2, which will give us sufficient conditions on the elements we need to add to prevent the reconstruction of the $\mathcal{RS}$ from the publicly available information. Thus, initializing the accumulator with $\texttt{NMWitnessesMax} + 1$ random elements where some of them satisfies the hypothesis of Theorem 2 will ultimately prevent any, even partial, successful execution of the Witness Forgery Attack [30].

**Proposition 1.** *Let $p \in \mathbb{N}$ be a prime such that $p - 1 = p_1^{e_1} \cdot ... \cdot p_n^{e_n}$ factorizes as the product of $n > 1$ powers of distinct primes $p_i \in \mathbb{N}$. Let $f(x) \in \mathbb{Z}/p\mathbb{Z}[x]$ be a polynomial with $n \leq t < p - 1$ distinct non-zero roots $x_1, \dots, x_t \in \mathbb{Z}/p\mathbb{Z}$ such that the multiplicative order in $(\mathbb{Z}/p\mathbb{Z})^*$ of $x_i$, for $1 \leq i \leq n$, is $p_i^{e_i}$. Then*

*i. The least $k > 0$ for which there exists $z \in \mathbb{Z}/p\mathbb{Z}$ and $g(x) \in \mathbb{Z}/p\mathbb{Z}[x]$ such that $g(x)f(x) \equiv x^k - z \mod p$ is $k = p - 1$.*

*ii. The degree of the minimal-degree non-constant monomial of $f(x)$ is $s$ with $0 < s < t$.*

*Proof.* Suppose there exists $z \in \mathbb{Z}/p\mathbb{Z}$ and $g(x) \in \mathbb{Z}/p\mathbb{Z}[x]$ such that

$$g(x)f(x) \equiv x^k - z \mod p$$

Then, each root $x_1, \dots, x_t$ of $f(x)$ must be a root for $x^k - z$ in $\mathbb{Z}/p\mathbb{Z}$, that is

$$x_1^k \equiv \dots \equiv x_t^k \equiv z \mod p \tag{1}$$

Since, by hypothesis $(\mathbb{Z}/p\mathbb{Z})^* \simeq \mathbb{Z}/(p-1)\mathbb{Z} \simeq \langle x_1 \rangle \times \dots \times \langle x_n \rangle$ with $n > 1$ we have $z \in \bigcap_{i=1}^{t} \langle x_i \rangle \leq \bigcap_{i=1}^{n} \langle x_i \rangle = \langle 1 \rangle$. Hence a solution to (1) exists only if $z \equiv 1$ and the least $k$ for which it holds is $k = lcm(\text{ord}(x_1), \dots, \text{ord}(x_t)) \geq lcm(\text{ord}(x_1), \dots, \text{ord}(x_n)) = p - 1$. Since $k \leq p - 1$, we have $k = p - 1$.

It follows that as long as $t < p - 1$, there are no $z \in \mathbb{Z}/p\mathbb{Z}$ such that $f(x) \equiv x^t - z \mod p$. Hence the degree of the minimal-degree non-constant monomial of $f(x)$ is $s$ with $0 < s < t$. □

**Theorem 2.** *Let $p \in \mathbb{N}$ be a prime such that $p - 1 = p_1^{e_1} \cdot ... \cdot p_n^{e_n}$ factorizes as the product of $n > 1$ powers of distinct primes $p_i \in \mathbb{N}$. Let $f(x) \in \mathbb{Z}/p\mathbb{Z}[x]$ be a polynomial with $n \leq t < p - 1$ distinct non-zero roots $x_1, \dots, x_t \in \mathbb{Z}/p\mathbb{Z}$ such that the multiplicative order in $(\mathbb{Z}/p\mathbb{Z})^*$ of $x_i$, for $1 \leq i \leq n$, is $p_i^{e_i}$.*

*For a secret $\alpha \in \mathbb{Z}/p\mathbb{Z}$, suppose $P \neq O \in G_1, V = f(\alpha)P$, $f(x)$ and the map $\varphi : \mathbb{N}_{<p-1} \to G_1$ defined as $\varphi(k) = \alpha^k V$ are known*[5].

*Then, from this information, it is not possible to obtain by using linear combinations any point of the form $\alpha^k P$ with $1 \leq k < p - 1$.*

*Proof.* Let $(\mathcal{V}, +)$ be the vector space of polynomials with degree lower equal $p - 2$ and $\mathcal{B} = \{1, x, \dots, x^{p-2}\}$ its $(p-1)-$dimensional canonical basis.

Since $\varphi(k) = \alpha^k f(\alpha)P$, the statement is then equivalent to prove that for every $1 \leq k < p - 1$

$$\text{rank}\left(\begin{bmatrix} 1 \\ f(x) \\ xf(x) \\ \vdots \\ x^{p-t-2}f(x) \\ x^k \end{bmatrix}_{\mathcal{B}}\right) = p - t + 1$$

[5] The knowledge of $\varphi$ models the fact that from the coordinates of $\Omega_i$ together with the knowledge of the sets $\mathcal{A}_i$ and $\mathcal{D}_i$ it could be possible to compute some values $g(\alpha)V_i$ for known polynomials $g(x) \in \mathbb{Z}/p\mathbb{Z}[x]$ and, ultimately, the values $\alpha^k V_i$ for $1 \leq k \leq \texttt{batchMax}$.

The rank is maximum when the row vectors are linearly independent in $\mathcal{V}$, that is for any $a_0, \dots, a_{p-t-2}, b, c \in \mathbb{Z}/p\mathbb{Z}$ such that

$$\left( \sum_{j=0}^{p-t-2} a_j x^j f(x) \right) + bx^k + c = 0 \tag{2}$$

we have $a_0 \equiv \cdots \equiv a_{p-t-2} \equiv b \equiv c \equiv 0$.

We will prove the statement by exhaustion on the values of $k$.

$\boxed{1 \leq k < t\text{:}}$ The dependence relation (2) can be rewritten as

$$g(x)f(x) = -bx^k - c$$

where $g(x) = \sum_{j=0}^{p-t-2} a_j x^j$. By hypothesis $f(x)$ has $t$ different roots, while $-bx^k - c$ can have at most $k < t$ distinct roots. The equation then holds only if both sides are equal to the 0 polynomial, that is $-bx^i - c = 0$ and $g(x) = 0$. This implies $a_0 \equiv \cdots \equiv a_{p-t-2} \equiv 0$ and $b \equiv c \equiv 0$ because the elements $\{1, x, \dots, x^{p-t-2}\}$ are linearly independent vectors of $\mathcal{V}$.

$\boxed{k = t\text{:}}$ In this case the dependence relation (2) can be rewritten as

$$g(x)f(x) = -bx^t - c$$

with $g(x)$ defined as in the previous case. By hypothesis $f(x)$ has $t$ distinct roots, while the right side can have at most $t$ distinct roots. This implies that $g(x) = g(0) = a_0$ is a constant polynomial or, equivalently, that $a_1 \equiv \cdots \equiv a_{p-t-2} \equiv 0$. Suppose by contradiction that $a_0 \neq 0$, then $f(x) = -a_0^{-1}bx^t - a_0^{-1}c$ is a contradiction since, by Proposition 1, the degree of the minimal-degree non-constant monomial of $f(x)$ is $s$ with $s \neq t$ and $s > 0$. Hence $a_0 \equiv 0$, and then $-bx^t - c = 0$ which implies $b \equiv c \equiv 0$.

$\boxed{t < k < p-1\text{:}}$ Let $k = t + k'$ with $1 \leq k' \leq p - t - 2$. From

$$g(x)f(x) = -bx^{t+k'} - c$$

it follows that $deg(g) \leq k'$ and then, if $k' < p - t - 2$, we have $a_{k'+1} \equiv \cdots \equiv a_{p-t-2} \equiv 0$.

Assume, by contradiction, $b \neq 0$ and $c \equiv 0$. In this case the dependence relation becomes $g(x)f(x) = -bx^{t+k'}$, but the right side has only 0 as root while the left side has, by hypothesis, at least $t$ non-zero distinct roots. This implies, similarly as before, that $g(x) = 0$ and $b \equiv 0$, a contradiction.

Let us therefore assume $b \neq 0$ and $c \neq 0$. In this case the dependence relation can be rewritten as $g'(x)f(x) = x^{t+k'} - z$ where $g'(x) = (-b)^{-1}g(x)$ and $z = (-b)^{-1}c \neq 0$.

If, by contradiction, $g'(x) \neq 0$, then, by Proposition 1, the least value for $t+k'$ such that the dependence relation holds is $t + k' = p - 1$, that is $k' = p - t - 1$, a contradiction to $1 \leq k' \leq p-t-2$. Hence $g'(x)=0$, which in turn implies $b \equiv c \equiv 0$, a contradiction to our assumption $b \neq 0$.

It follows that $b \equiv 0$ and then $g(x)f(x) = -c$. Since by hypothesis $f$ has $t$ distinct roots, this equation holds only if $c=0$ and $g(x)=0$, which, similarly as before, implies $a_0 \equiv \cdots \equiv a_{k'} \equiv b \equiv c \equiv 0$. □

We are now ready to explicitly define the Accumulator Initialization procedure for our protocol:

**Accumulator Initialization** Set an upper limit `NMWitnessesMax` to the total number of issuable non-membership witnesses. Assume $p$ is such that $p-1 = p_1^{e_1} \cdot \ldots \cdot p_n^{e_n}$ factorizes as the product of $n > 1$ powers of distinct primes $p_i$ and consider $n$ elements $x_1, \ldots, x_n \in \mathbb{Z}/p\mathbb{Z}$ such that the multiplicative order in $(\mathbb{Z}/p\mathbb{Z})^*$ of $x_i$ is $p_i^{e_i}$, for $1 \leq i \leq n$. Then, the Accumulator Manager sets

$$\mathcal{Y}_{V_0} = \{x_1, \ldots, x_n\} \cup \{\texttt{NMWitnessesMax} - n + 1 \text{ random elements in } \mathcal{ACC}\}$$

so that $|\mathcal{Y}_{V_0}| = \texttt{NMWitnessesMax} + 1$ and defines the corresponding initialization polynomial as $f_0(x) = \prod_{x_i \in \mathcal{Y}_{V_0}} (x - x_i)$, where $V_0 = f_0(\alpha)P$. He then publishes $(V_0, \emptyset)$, the accumulator state at epoch 0, and keeps secret and never deletes the elements in $\mathcal{Y}_{V_0}$.

We note that as soon as an epoch changes, the Accumulator Manager publishes the corresponding Batch Witnesses Update information: at epoch 1, for example, this corresponds to the new state $(V_1, \mathcal{Y}_{V_1})$, the updating vector $\Omega_1$ and the polynomials $d_{\mathcal{A}_1}(x)$ and $d_{\mathcal{D}_1}(x)$. At this point, the polynomial

$$f_{V_1}(x) = f_0(x) \cdot f_1(x) = \prod_{y_i \in \mathcal{Y}_{V_0}} (y_i + x) \cdot \prod_{y_j \in \mathcal{Y}_{V_1}} (y_j + x)$$

has $|\mathcal{Y}_{V_0}| + |\mathcal{Y}_{V_1}|$ distinct non-zero roots, $n$ of which are $x_1, \ldots, x_n$, and is such that $V_1 = f_{V_1}(\alpha)P$. Even if it is possible to obtain from $\Omega_1$, $\mathcal{A}_1$ and $\mathcal{D}_1$ all the values $\alpha^k V_1$ for $1 \leq k < p-1$ (we relax the condition $k \leq \texttt{batchMax}$) we still are under the hypothesis of Theorem 2, which assures us the infeasibility to obtain any element of the $\mathcal{RS}$. This reasoning can be easily generalized to any subsequent epoch.

One last question arises with regard to all these considerations: is it possible to obtain some elements in the $\mathcal{RS}$ combining the vectors $\Omega_i$ coming from different epochs? When the accumulator is initialized as described in Theorem 2, the answer is *no.* To show this, consider, without loss of generality, the $m$ vectors $\Omega_1, \ldots, \Omega_m$, where the $j$-entry of any $\Omega_i$ is of the form $c_j V_i$. Hence a linear combination with coefficients $a_{i,j} \in \mathbb{Z}/p\mathbb{Z}$ of entries of these vectors can be written as

$$\sum_{i=1}^{m} \sum_{j=0}^{|\texttt{batchMax}|} a_{i,j} c_j V_i = \left( \sum_{i=1}^{m} \sum_{j=0}^{|\texttt{batchMax}|} a_{i,j} c_j f_i(\alpha) \right) \cdot f_0(\alpha) P = g(\alpha) \cdot V_0$$

where $g(x) = \sum_{i=1}^{m}\sum_{j=0}^{\texttt{batchMax}} a_{i,j}c_j f_i(x)$. In other words, in the luckiest situation, what we can obtain combining all these vectors is not more than the elements $\alpha^k V_0$ with $1 \leq k \leq \texttt{batchMax}$ which, as we already discussed, does not permit to obtain any element in $\mathcal{RS}$.

## 8 Zero-Knowledge Proof of Knowledge

We now explicitly show how an interactive zero-knowledge protocol can be instantiated between a Prover and a Verifier to prove the ownership of a valid non-membership witness $\bar{w}_{y,V}$ for $y$ with respect to the accumulator state $(V, \mathcal{Y}_V)$ (the corresponding protocol to show ownership of a valid membership witness $w_{y,V}$ is similar and will not be discussed).

We will extend the zero-knowledge proof of knowledge protocol defined by Boneh *et al.* in [10], which proves under the Decision Linear Diffie-Hellman assumption the knowledge of a pair $(y, C)$ such that $(y + \alpha)C = V$, in order to support tuples $(y, C, d)$ which verify $(y + \alpha)C + dP = V$.

However, for non-membership witnesses we need to further ensure that $d \neq 0$ or, equivalently in $G_1$, that has a multiplicative inverse. At this scope we will then consider, for a random generator $K \in G_1$ and random $a, b \in \mathbb{Z}/p\mathbb{Z}$, the Pedersen commitments $E_d = dP + aK$ and $E_{d^{-1}} = d^{-1}P + bK$ for $d$ and $d^{-1}$, respectively. Noticing that $P = dE_{d^{-1}} - dbK$, we then extend the protocol applying EQ-composition to the factor $d$ among the values $E_d$ and $P$, thus showing that $E_{d^{-1}}$ is a Pedersen commitment to the multiplicative inverse of the committed value in $E_d$.

We refer to [10] and [29] for the completeness, soundness and (special) honest-verifier zero-knowledgeness proof of Boneh et al. protocol and EQ-composition for multiplicative-inverse relation, respectively.

Under the Random Oracle Model, we can made such proof of knowledge non-interactive and full zero-knowledge by applying Fiat-Shamir heuristic [1]. We will do so by using an heuristic variant adopted by Boneh *et al.* in [10] in order to reduce Prover's proof size. We assume that calls to the random oracle can be concretely realized through evaluations to a cryptographic hash function $H : \{0, 1\}^* \to \mathbb{Z}/p\mathbb{Z}$.

The resulting protocol is the following.

**Setup** The Prover and Verifier agree on the public values $P, X, Y, Z, K \in G_1$ and $\tilde{P}, \tilde{Q} \in G_2$, where $X, Y, Z, K$ are distinct random generators of $G_1$.

**Proof Of Knowledge** The Prover randomly selects $\sigma, \rho, \tau, \pi \in \mathbb{Z}/p\mathbb{Z}$ and computes

$$E_C = C + (\sigma + \rho)Z, \qquad E_d = dP + \tau K, \qquad E_{d^{-1}} = d^{-1}P + \pi K,$$

$$T_\sigma = \sigma X, \qquad T_\rho = \rho Y, \qquad \delta_\sigma = y\sigma, \qquad \delta_\rho = y\rho$$

A non-interactive zero knowledge Proof of Knowledge of values $(y, d, \sigma, \rho, \tau, \pi, \delta_\sigma, \delta_\rho)$ satisfying

$$P = dE_{d^{-1}} - d\pi K, \qquad E_d = dP + \tau K,$$

$$\sigma X = T_\sigma, \qquad \rho Y = T_\rho, \qquad yT_\sigma - \delta_\sigma X = O, \qquad yT_\rho - \delta_\rho Y = O,$$

$$e(E_C, \tilde{P})^y e(Z, \tilde{P})^{-\delta_\sigma - \delta_\rho} e(Z, \tilde{Q})^{-\sigma-\rho} e(K, \tilde{P})^{-\tau} = \frac{e(V, \tilde{P})}{e(E_C, \tilde{Q}) e(E_d, \tilde{P})}$$

is undertaken between Prover and Verifier as follows:

**Blinding (P)** The Prover randomly picks $r_y$, $r_u$, $r_v$, $r_w$, $r_\sigma$, $r_\rho$, $r_{\delta_\sigma}$, $r_{\delta_\rho} \in \mathbb{Z}/p\mathbb{Z}$, computes

$$R_A = r_u P + r_v K, \qquad R_B = r_u E_{d^{-1}} + r_w K,$$

$$R_E = e(E_C, \tilde{P})^{r_y} e(Z, \tilde{P})^{-r_{\delta_\sigma} - r_{\delta_\rho}} e(Z, \tilde{Q})^{-r_\sigma - r_\rho} e(K, \tilde{P})^{-r_v},$$

$$R_\sigma = r_\sigma X, \qquad R_\rho = r_\rho Y, \qquad R_{\delta_\sigma} = r_y T_\sigma - r_{\delta_\sigma} X, \qquad R_{\delta_\rho} = r_y T_\rho - r_{\delta_\rho} Y$$

**Challenge (P)** The Prover sets the challenge $c \in \mathbb{Z}/p\mathbb{Z}$ to

$$c = H(V, E_C, E_d, E_{d^{-1}}, T_\sigma, T_\rho, R_A, R_B, R_E, R_\sigma, R_\rho, R_{\delta_\sigma}, R_{\delta_\rho})$$

**Response (P)** The Prover computes

$$s_y = r_y + cy, \qquad s_u = r_u + cd, \qquad s_v = r_v + c\tau, \qquad s_w = r_w - cd\pi,$$

$$s_\sigma = r_\sigma + c\sigma, \qquad s_\rho = r_\rho + c\rho, \qquad s_{\delta_\sigma} = r_{\delta_\sigma} + c\delta_\sigma, \qquad s_{\delta_\rho} = r_{\delta_\rho} + c\delta_\rho$$

and sends $(E_C, E_d, E_{d^{-1}}, T_\sigma, T_\rho, c, s_y, s_u, s_v, s_w, s_\sigma, s_\rho, s_{\delta_\sigma}, s_{\delta_\rho})$ to the Verifier.

**Verify (V)** The Verifier computes

$$R_A = s_u P + s_v K - cE_d, \qquad R_B = s_w K + s_u E_{d^{-1}} - cP,$$

$$R_\sigma = s_\sigma X - cT_\sigma, \qquad R_\rho = s_\rho Y - cT_\rho, \qquad R_{\delta_\sigma} = s_y T_\sigma - s_{\delta_\sigma} X, \qquad R_{\delta_\rho} = s_y T_\rho - s_{\delta_\rho} Y,$$

$$R_E = e(E_C, \tilde{P})^{s_y} \cdot e(Z, \tilde{P})^{-s_{\delta_\sigma} - s_{\delta_\rho}} \cdot e(Z, \tilde{Q})^{-s_\sigma - s_\rho} \cdot e(K, \tilde{P})^{-s_v} \cdot \left( \frac{e(V, \tilde{P})}{e(E_C, \tilde{Q}) e(E_d, \tilde{P})} \right)^{-c}$$

and accepts if $c = H(V, E_C, E_d, E_{d^{-1}}, T_\sigma, T_\rho, R_A, R_B, R_E, R_\sigma, R_\rho, R_{\delta_\sigma}, R_{\delta_\rho})$.

**Complexity Analysis.** Within this protocol, zero-knowledge proofs for valid non-membership witnesses consists of 5 elements in $G_1$ and 11 elements in $\mathbb{Z}/p\mathbb{Z}$, while they consists of 3 elements in $G_1$ and 6 elements in $\mathbb{Z}/p\mathbb{Z}$ in the case of membership witnesses. Thus, if elliptic curve points compression is used, zero-knowledge non-membership and membership proofs can be represented with $5(\log q + 1) + 9 \log p$ bits and $3(\log q + 1) + 6 \log p$ bits, respectively. In our concrete instantiation (see Section 9) this translates to 4926 bits $\approx$ 616 bytes proofs for non-membership witnesses and 3135 bits $\approx$ 392 bytes proofs for membership witnesses.

As regards computational costs, if the quantities $e(Z,\tilde{P})$, $e(Z,\tilde{Q})$, $e(K,\tilde{P})$ and $e(V,\tilde{P})$ are pre-computed and stored by both Prover and Verifier, zero-knowledge proofs of knowledge for non-membership witnesses are computed with 15 scalar-point multiplications in $G_1$, 7 point additions in $G_1$, 4 exponentiation in $G_T$ and 1 pairing. We note that the Prover can reduce the cost of evaluating $e(E_C,\tilde{P})$ by computing and storing the value $e(C,\tilde{P})$. Thus, with just 1 pairing per-epoch, the Prover can compute each $e(E_C,\tilde{P})$ as $e(Z,\tilde{P})^{\sigma+\rho} \cdot e(C,\tilde{P})$ with 1 exponentiation and 1 multiplication in $G_T$. Using this optimization, the cost to compute a proof of knowledge of a membership witness boils down to a total of 9 scalar-point multiplications in $G_1$, 3 point additions in $G_1$, 5 exponentiation in $G_T$ and 1 multiplication in $G_T$.

Similarly, the Verifier needs 16 scalar-point multiplications in $G_1$, 9 point additions in $G_1$, 4 exponentiation in $G_T$ and 2 pairings (by merging the term $e(E_C,\tilde{P})^{s_y} e(E_C, c\tilde{Q})$ in $e(E_C, s_y\tilde{P} + c\tilde{Q})$) to verify a non-membership witness zero-knowledge proof, while he needs 10 scalar-point multiplications in $G_1$, 5 point additions in $G_1$, 3 exponentiation in $G_T$ and 1 pairing to verify a zero-knowledge proof of knowledge for a membership witness.

## 9 Implementation Results

To show efficiency and its practical relevance, we implemented the proposed accumulator scheme by using RELIC library [31]. In order to guarantee a security level of 128-bits, we selected the available pairing-friendly Type-III prime curve *B12-P446*. We then benchmarked the main features of the proposed accumulator, obtaining the following average results:

- **Accumulator Updates**: 1.27 seconds to add 1.000.000 elements (random elements generation included); 0.48 seconds to delete 1.000.000 elements.
- **Witness Issuing**: 1.9 milliseconds to issue a membership witness; 229.5 milliseconds for a non-membership witness (1.000.000 elements accumulated).
- **Witness Verification**: 2.2 milliseconds to verify a membership witness; 3.2 milliseconds to verify a non-membership witness.
- **Public Batch Witness Update**: 37.9 seconds to generate batch update data corresponding to a 10.000 elements batch addition operation; 27.1 seconds for a 10.000 elements batch deletion operation.
- **Batch Witness Update**: 2.12 seconds to update either membership or non-membership witness after a batch addition or deletion operation of 10.000 elements.
- **Zero-Knowledge Proof Creation**: 5.2 milliseconds to create a zero knowledge proof of knowledge of a membership witness; 7.4 milliseconds to create a proof for a non-membership witness.
- **Zero-Knowledge Proof Verification**: 6.5 milliseconds to verify a zero knowledge proof of knowledge of a membership witness; 11.2 milliseconds to verify a proof for a non-membership witness.

These benchmarks came from running our implementation on a standard Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz desktop provided with 8.00GB of RAM and running Ubuntu 18.04 x64. No parallelization was used.

## 10 Conclusions

We presented a Dynamic Universal Accumulator in the Accumulator Manager setting for efficient Type-III pairing-friendly elliptic curves, which supports batch operations and batch membership and non-membership public witness update.

The proposed accumulator extends a combination of previous schemes by adding batch operations, enabling users to update witnesses in optimal time. Furthermore, since batch update data is designed to be decoupled from users' witnesses, our protocol permits (privacy-preserving) witness updates delegation, thus enabling lightweight users to keep their witnesses updated with a constant number of elementary operations.

We further showed how to initialize the accumulator in order to be safe from an attack which would allow to forge witnesses for non-authorized elements, an essential requirement in the case the accumulator scheme is used as an authentication mechanism under the Accumulator Manager authority.

We then described how to instantiate a zero-knowledge proof of ownership of a valid witness for a given accumulator state and we implemented the accumulator logic along with batch operations, the public witness update protocol and the zero-knowledge proof mechanism in order to show its practical relevance as an efficient and scalable privacy-preserving authentication mechanism.

## References

1. Fiat, A. and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. *In Conference on the Theory and Application of Cryptographic Techniques*, 186–194 (1986)
2. Merkle, R. C. A certified digital signature. *In Advances in Cryptology – CRYPTO 1989*, 218-–238 (1989)
3. Benaloh, J. and de Mare, M. One-way Accumulators: A Decentralized Alternative to Digital Signatures. *In EUROCRYPT*, 274–285 (1993)
4. Baric, N. and Pfitzmann, B. Collision-free Accumulators and Fail-stop Signature Schemes Without Trees. *In EUROCRYPT*, 480–494, (1997)
5. Sander, T. Efficient Accumulators Without Trapdoor. *In ICICS*, 252–262 (1999)
6. Buldas, A., Laud, P. and Lipmaa, H. Accountable Certificate Management Using Undeniable Attestations. *In ACM CCS*, 9–17 (2000)
7. Chang, Y.-C. and Lu, C.-J. Oblivious Polynomial Evaluation and Oblivious Neural Learning. *In Advances in Cryptology — ASIACRYPT 2001*, 369–384 (2001)
8. Buldas, A., Laud, P. and Lipmaa, H. Eliminating Counterevidence with Applications to Accountable Certificate Management. *Journal of Computer Security*, **10:2002**, (2002)
9. Camenisch, J. and Lysyanskaya, A. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. *In CRYPTO*, 61–76 (2002)

10. Boneh, D., Boyen, X., Shacham, H. Short Group Signatures. *In Advances in Cryptology – CRYPTO 2004*, Springer LNCS, 41–55 (2004)
11. Nguyen, L. Accumulators from Bilinear Pairings and Applications. *In CT-RSA*, Springer LNCS, **3376**, 275–292 (2005)
12. Naor, M. and Pinkas, B. Oblivious polynomial evaluation. *Siam Journal On Computing.* **35**, 1254–1281 (2006)
13. Li, J., Li, N. and Xue, R. Universal Accumulators with Efficient Nonmembership Proofs. *In ACNS*, Springer LNCS, **4521**, 253–269 (2007)
14. Boneh, D. and Boyen, X. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, **21(2)**, 149–177, (2008).
15. Camacho, P., Hevia, A., Kiwi, M. A. and Opazo, R. Strong Accumulators from Collision-Resistant Hashing. *In ISC*, Springer LNCS, **4222**, 471–486 (2008)
16. Damgård, I. and Triandopoulos, N. Supporting Non-membership Proofs with Bilinear-map Accumulators. *IACR Cryptology ePrint Archive*, 538 (2008)
17. Galbraith, S. D., Paterson, K. G. and Smart, N. P. Pairings for cryptographers. Discrete Applied Mathematics, **156(16)**, 3113–3121 (2008).
18. Au, M. H., Tsang, P. P., Susilo, W. and Mu, Y. Dynamic Universal Accumulators for DDH Groups and Their Application to Attribute-Based Anonymous Credential Systems. *In CT-RSA*, Springer LNCS, **5473**, 295–308 (2009)
19. Camenisch, J. and Soriente, C. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. *In PKC 2009*, Springer LNCS, **5443**, 481–500 (2009)
20. Camacho, P. and Hevia, A. On the impossibility of batch update for cryptographic accumulators. *In International Conference on Cryptology and Information Security in Latin America*, 178–188 (2010)
21. Aranha, D. F., Fuentes-Castañeda, L., Knapp, E., Menezes, A. and Rodríguez-Henríquez, F. Implementing pairings at the 192-bit security level. *In International Conference on Pairing-Based Cryptography*, 177–195 (2012)
22. Lipmaa, H. Secure Accumulators from Euclidean Rings without Trusted Setup. *In ACNS*, Springer LNCS, **7341**, 224–240 (2012)
23. Adj, G., Menezes, A., Oliveira, T. and Rodríguez-Henríquez, F. Computing Discrete Logarithms in $\mathbb{F}_{3^{6\cdot 137}}$ and $\mathbb{F}_{3^{6\cdot 163}}$ Using Magma. *In International Workshop on the Arithmetic of Finite Fields*, 3–22 (2014)
24. Granger, R., Kleinjung, T. and Zumbrägel, J. Discrete logarithms in the Jacobian of a genus 2 supersingular curve over GF($2^{367}$). *In NMBRTHRY list*, **30**, (2014)
25. Boneh, D. and Corrigan-Gibbs, H. Bivariate polynomials modulo composites and their applications. *In International Conference on the Theory and Application of Cryptology and Information Security*, 42–62 (2014)
26. Boneh, D., Bünz, B. and Fisch, B. Batching techniques for accumulators with applications to IOPs and stateless blockchains. *Cryptology ePrint Archive*, Report 2018/1188, (2018)
27. Hazay, C. Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs. *Journal Of Cryptology.* **31**, 537–586 (2018)
28. Kleinjung, T., and Wesolowski, B. Discrete logarithms in quasi-polynomial time in finite fields of fixed characteristic. *IACR Cryptology ePrint Archive*, 751 (2019)
29. Schoenmakers, B. Cryptographic Protocols. Lecture Notes, 108–109 (2019)
30. Biryukov, A., Udovenko, A. and Vitto, G. Cryptanalysis of Au et al. Dynamic Universal Accumulator. *IACR Cryptology ePrint Archive*, 598 (2020)
31. Aranha, D. F. and Gouvêa C. P. L. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`