Cryptographic Accumulators: New Definitions, Enhanced Security, and Delegatable Proofs

Anaïs Barthoulot[1], Olivier Blazy[2], and Sébastien Canard[3]

Université de Montpellier, LIRMM, Montpellier, France
anais.barthoulot@lirmm.fr
École Polytechnique, Palaiseau, France
olivier.blazy@polytechnique.edu
Télécom Paris, Palaiseau, France
sebastien.canard@telecom-paris.fr

Abstract. Cryptographic accumulators, introduced in 1993 by Benaloh and De Mare, represent a set with a concise value and offer proofs of (non-)membership. Accumulators have evolved, becoming essential in anonymous credentials, e-cash, and blockchain applications. Various properties like *dynamic* and *universal* emerged for specific needs, leading to multiple accumulator definitions. In 2015, Derler, Hanser, and Slamanig proposed a unified model, but new properties, including *zero-knowledge* security, have arisen since. We offer a new definition of accumulators, based on Derler *et al.*'s, that is suitable for all properties. We also introduce a new security property, *unforgeability of private evaluation*, to protect accumulator from forgery and we verify this property in Barthoulot, Blazy, and Canard's recent accumulator. Finally we provide discussions on security properties of accumulators and on the delegatable (non-)membership proofs property.

Keywords: Cryptographic accumulators · Dual pairing vector spaces · Security reductions

1 Introduction

Cryptographic accumulators. In 1993, Benaloh and De Mare [15] introduced the concept of a one-way accumulator as a family of one-way hash functions satisfying the *quasi-commutative* property. Later, Baric and Pfitzmann [12] extended the definition, characterizing accumulators as schemes enabling the concise representation, termed the accumulator, of a finite set of values. A cryptographic accumulator provides membership proofs for elements in the set. Some accumulators require an additional element, called the *witness*, for generating a membership proof, classifying them as *asymmetric* accumulators; those without this requirement are *symmetric* accumulators. This work focuses on asymmetric accumulators, comprising four algorithms $\mathsf{Setup}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{Verify}$. Specifically, Setup establishes accumulator parameters, Eval accumulates values, $\mathsf{WitCreate}$ generates membership witnesses, and Verify verifies membership. As for security, accumulators must satisfy the *collision resistance* property, preventing adversaries from finding an element not in the set and producing a fraudulent witness for this element.

Uniform Modeling. From this basic property, the literature has been very prolific in terms of additional functionalities and/or properties. We can mention the *dynamic* or the *universal* properties, which respectively allow the addition or removal of elements to the accumulated set and efficient updates to witnesses, as well as the generation of witnesses to prove the absence of certain elements. Most of these new properties were introduced to satisfy a specific (and sometimes unique) need, thus they do not help in having a global picture of cryptographic accumulators and their properties, with some rare exceptions [32,30,7]. In the asymmetric setting, we consider that the most relevant paper on such issues is the one by Derler *et al.* [30] in 2015, who proposed a uniform model for dynamic universal cryptographic accumulators, regrouping existing properties. We use this paper model's in this work. Recently, [9] proposed the first universally composable treatment of cryptographic accumulators. However, due to its limited adoption in the literature, we opt for a property-based definition of accumulators rather than a universally composable treatment.

Instantiations and Related Primitives. Since their introduction, several accumulators were built. The original ones were based on the RSA assumption [15,12], and many works used variants of it to build their own scheme [22,64,31]. In [54], Nguyen proposed an accumulator based on pairings, and others followed [5,28,21,4,35, ...]. Recently, a few works presented accumulators based on lattices, such as [50,43,57] and [53,6] proposed a code-based scheme. We can then divide existing accumulator instantiations in five categories: hash-based accumulators, lattice-based accumulators, pairing-based, code-based, and number theoretic accumulators. Several works studied the relations between cryptographic accumulators and other primitives: [35] showed that zero-knowledge sets implies zero-knowledge accumulators, which itself implies primary-secondary-resolver membership proof systems, [25] proved that vector commitment can be used to build dynamic accumulators, and later [45] proved that functional commitments for linear functions implies cryptographic accumulators with large universe (i.e., domain size can be exponential in the security parameter).

Applications. Cryptographic accumulators are versatile tools with diverse applications. Originally used for timestamping and membership testing [15], they have found utility in various areas such as fail-stop signatures [12], ID-based ring signatures [54], and distributed public key infrastructure [60]. However, their central focus today is on protecting individual privacy, especially in membership revocation for group signatures, direct anonymous attestations [22], and anonymous credentials [2,4]. Accumulators play a crucial role in authenticated data structures, addressing the challenge of authenticating set operations [58,45,35]. Operations on sets, including subset and disjointness, can be directly performed on sets represented by an accumulator [63]. Additionally, accumulator-based representations support fundamental set operations such as union, intersection, and set difference [58,35]. Furthermore, in the context of blockchain and digital cash systems, accumulators serve crucial roles. In blockchain, they streamline transaction verification by proving membership in valid transactions while preserving privacy. They also contribute to data compression in blockchain states, reducing storage needs and improving scalability. In digital cash systems, accumulators provide efficient methods for verifying transaction validity while safeguarding user anonymity [5,24].

As a result, they play an indispensable role in ensuring both security and privacy in digital financial transactions across various applications.

Our contributions. In this work, we present a new definition of cryptographic accumulators along with an overview of accumulator properties and features,that complements [30]'s model. Additionally, we engage in several discussions on accumulators properties. Specifically:

- We introduce a novel definition of cryptographic accumulators, representing the primary contribution of this work. For the sake of clarity, our scheme is static (*i.e.* not dynamic) and non-universal, but a definition for a dynamic universal scheme, as presented in [30], can easily be derived. This new definition offers flexibility as it serves as a foundational framework to incorporate all existing properties and functionalities of accumulators, thereby extending the model proposed by [30]. Furthermore, it aims to establish a standard in the field of cryptographic accumulators, thereby providing a unified framework for describing accumulators and their properties. To demonstrate the usability of our definition, we present a (informal) comprehensive and up-to-date overview of all accumulator properties, along with a state of the art, in Section 2.
- In Section 3, we focus on security properties of accumulators. First, we delve into a discussion on the property called *undeniability* within a specific scenario known as the trusted setup model. Then, we engage a discussion on a recently introduced security property, *obliviousness*. Finally, we explore the existing relations between these properties and establish new relations.
- In Section 4, we present the second contribution of this work: the introduction of a novel security property, called *unforgeability of private evaluation*. This property states that the scheme is resistant to attempts to forge or create false accumulators using the secret key. We also establish that the recent accumulator proposed in [14] satisfies this new security property. The latter operates in the asymmetric bilinear setting and employs dual pairing vector spaces [56], and to demonstrate that this scheme satisfies our new security property, we introduce a novel security assumption named *fixed argument dual pairing vector spaces inversion*. This assumption, constituting an auxiliary contribution, can be reduced to *computational Diffie Hellman* assumption and represents the first computational assumption for dual pairing vector spaces and may hold independent significance for future works.
- We conclude this paper with a discussion on a property known as delegatable (non-)membership proofs. In greater detail, we explore the requirements necessary to achieve this property with the aim of presenting a generic construction.

2 Cryptographic Accumulators

In this section, we formally introduce cryptographic accumulators and provide an exhaustive list of definitions, functionalities, and properties associated with this primitive. In line with our paper's introduction, we emphasize modern accumulator definitions over the original one by Benaloh and De Mare [15]. Throughout the remainder of our work, we focus on asymmetric accumulators, with the understanding that many of the

properties apply to symmetric accumulators as well. We opt for asymmetric accumulators due to their improved efficiency: in the accumulators literature, it is admitted that symmetric accumulators cannot have a size less than linear in the number of accumulated elements, while asymmetric schemes can produce accumulators of constant size.

2.1 Our New Definition

We propose a definition of accumulators based on the definition given by Derler *et al.* [30] (that we slightly simplify), and based on proof systems as the definition given by Acar and Nguyen [1]. The motivation behind introducing this new definition stems from the absence of a sufficiently modular definition that can adapt to various properties. Currently, defining an accumulator scheme for a specific property requires a tailored approach, resulting in the need to redefine the accumulator to align with specific requirements. Our proposed definition addresses this limitation, offering a modular framework that can be applied universally across different properties. We start by giving the definition of proof system and its associated properties.

Definition 1. ***Proof System [1].*** *Let $\mathcal{R}$ be an efficiently computable relation of* $(\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit})$ *with setup parameters* Para*, a statement* Sta*, and a witness* Wit*. A non-interactive proof system for $\mathcal{R}$ consists of 3 PPT algorithms: a* Setup*, a prover* Prove*, and a verifier* Verif*. A non-interactive proof system* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$ *must be* complete *and* sound*. Completeness means that for every PPT adversary $\mathcal{A}$, the following is negligible*

$$\left|\Pr\left[\begin{array}{c}\mathsf{Para} \leftarrow \mathsf{Setup}(\lambda); (\mathsf{Sta}, \mathsf{Wit}) \leftarrow \mathcal{A}(\mathsf{Para}); \mathsf{Proof} \leftarrow \mathsf{Prove}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit}): \\ \mathsf{Verif}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Proof}) = 1 \text{ if } (\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit}) \in \mathcal{R}\end{array}\right] - 1\right|.$$

Soundness means that for every PPT adversary $\mathcal{A}$, the following is negligible

$$\left|\Pr\left[\begin{array}{c}\mathsf{Para} \leftarrow \mathsf{Setup}(\lambda); (\mathsf{Sta}, \mathsf{Proof}) \leftarrow \mathcal{A}(\mathsf{Para}): \\ \mathsf{Verif}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Proof}) = 0 \text{ if } (\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit}) \notin \mathcal{R}, \forall \mathsf{Wit}\end{array}\right] - 1\right|.$$

We now present our new definition of accumulators, based on proof systems.

Definition 2. ***Cryptographic Accumulator [15,30].*** *A cryptographic accumulator scheme is a tuple of efficient algorithms defined as follows:*

- $\mathsf{Gen}(\lambda)$*: the generation algorithm takes as input a security parameter λ. It returns a key pair $\mathfrak{K} = (\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$, where $\mathsf{pk}_{\mathsf{acc}}$ contains the setup parameters* Para *of $\mathcal{R}$, an efficiently computable relation.* Gen *can also be seen as the* Setup *algorithm of a proof system* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$ *for $\mathcal{R}$.*
- $\mathsf{Eval}(\mathfrak{K}, \mathcal{X})$*: the evaluation algorithm takes as input the accumulator key pair $\mathfrak{K}$ and a set $\mathcal{X}$ to be accumulated. It returns an accumulator $\mathsf{acc}_{\mathcal{X}}$ together with some auxiliary information* aux*. Notice that* $(\mathsf{acc}_{\mathcal{X}}, \mathsf{aux})$ *form a statement* Sta *for $\mathcal{R}$.*
- $\mathsf{WitCreate}(\mathfrak{K}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x)$*: the witness creation algorithm takes as input the accumulator key pair $\mathfrak{K}$, an accumulator $\mathsf{acc}_{\mathcal{X}}$, the associated set $\mathcal{X}$, auxiliary information* aux*, and an element x. If $x \in \mathcal{X}$ it outputs a witness $\mathsf{wit}_x^{\mathcal{X}}$, otherwise it outputs a reject symbol $\perp$. Note that $\mathsf{wit}_x^{\mathcal{X}}$ forms a witness* Wit *for $\mathcal{R}$.*

– $\mathsf{CompProof}(\mathsf{pk_{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}_x^{\mathcal{X}}, x)$*: the proof computation algorithm takes as input the accumulator public key* $\mathsf{pk_{acc}}$ *that contains the proof system parameters* Para*, an accumulator* $\mathsf{acc}_{\mathcal{X}} = \mathsf{Sta}$ *and associated auxiliary information, a witness* $\mathsf{wit}_x^{\mathcal{X}} = \mathsf{Wit}$*, and element* x*. It runs the proof system prover algorithm* $\mathsf{Prove}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit})$ *and outputs the result* Proof.
– $\mathsf{Verify}(\mathsf{pk_{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{Proof})$*: the verification algorithm takes as input the accumulator public key* $\mathsf{pk_{acc}}$ *that contains the proof system parameters* Para*, an accumulator* $\mathsf{acc}_{\mathcal{X}} = \mathsf{Sta}$ *and associated auxiliary information* aux*, and a proof* Proof*. It runs the proof system verification algorithm* $\mathsf{Verif}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Proof})$*. If* $(\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit}) \in \mathcal{R}$ *(meaning that* $\mathsf{wit}_x^{\mathcal{X}}$ *is correct and thus that* $x \in \mathcal{X}$*) it returns* 1*, otherwise it returns* 0*.*

Note 1. The algorithm Gen is ran by a third party, called sometimes accumulator manager. We will come back on the trust of this party in Section 2.2.

We now present the two fundamental properties of accumulators, *correctness* and *collision resistance*. We start with the former that states that for all honestly generated keys, computed accumulators and witnesses, the Verify algorithm always return 1.

Definition 3. ***Correctness.*** *A cryptographic accumulator is said to be* correct *if for all security parameter* λ*, all set of values* $\mathcal{X}$*, and all element* x *such that* $x \in \mathcal{X}$*:*

$$\Pr\left[\begin{array}{c} \mathfrak{K} = (\mathsf{sk_{acc}}, \mathsf{pk_{acc}}) \leftarrow \mathsf{Gen}(\lambda), (\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}) \leftarrow \mathsf{Eval}(\mathfrak{K}, \mathcal{X}), \\ \mathsf{wit}_x^{\mathcal{X}} \leftarrow \mathsf{WitCreate}(\mathfrak{K}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x) \\ \mathsf{Proof} \leftarrow \mathsf{CompProof}(\mathsf{pk_{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}_x^{\mathcal{X}}, x) : \\ \mathsf{Verify}(\mathsf{pk_{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}) = 1 \end{array}\right] = 1$$

Regarding security of cryptographic accumulators, several notions were introduced such as *undeniability* [47], *indistinguishability* [30] or *zero-knowledge* [35] for example. We here only formally present the property of *collision resistance*, but in Section 2.2 we give an exhaustive list of all accumulator security properties and a (informal) definition for all of them. Informally a cryptographic accumulator is said to be *collision resistant* if it is hard for an adversary to forge a witness for an element that is not in the accumulated set.

Definition 4. ***Collision resistance [12,30].*** *An accumulator scheme is said to satisfy* collision resistance[1] *if all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}_{\mathcal{A}}^{CR}(\lambda) := \Pr\left[\begin{array}{c} \mathfrak{K} = (\mathsf{sk_{acc}}, \mathsf{pk_{acc}}) \leftarrow \mathsf{Gen}(\lambda), (\mathcal{X}, \mathsf{wit}_x^{\mathcal{X}}, x) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk_{acc}}) \\ (\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}) \leftarrow \mathsf{Eval}(\mathfrak{K}, \mathcal{X}) \\ \mathsf{Proof} \leftarrow \mathsf{CompProof}(\mathsf{pk_{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}_x^{\mathcal{X}}, x) : \\ \mathsf{Verify}(\mathsf{pk_{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}) = 1 \wedge x \notin \mathcal{X} \end{array}\right]$$

where $\mathcal{O} = \left\{\mathcal{O}^E, \mathcal{O}^W\right\}$ *and* $\mathcal{O}^E, \mathcal{O}^W$ *represent the oracles for the algorithms* Eval *and* WitCreate *respectively. An adversary is allowed to query them an arbitrary number of times. Note that if* $\mathcal{O}^W$ *is queried for an element that does not belong to the accumulator also provided as input, the oracle outputs a reject symbol.*

[1] In some works, "collision resistance" is called "collision freeness", "soundness" or "set binding" [25]. We will only use the terms *collision resistance* in the following.

The following theorem establishes that correctness and collision resistance of the accumulator scheme hold if completeness and soundness of the underlying proof system hold respectively.

Theorem 1. *If the proof system* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$ *is respectively* complete *and* sound*, then the accumulator* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{CompProof}, \mathsf{Verify})$ *is respectively* correct *and satisfies* collision resistance.

We prove the theorem in two steps, corresponding to the following lemmas.

Lemma 1. *If the proof system* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$ *is* complete*, then the accumulator* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{CompProof}, \mathsf{Verify})$ *is* correct*.*

Proof. First, let us see the correctness property as a game between a challenger and an adversary. The aim of the adversary is to find a set $\mathcal{X}$ and an element x such that $x \in \mathcal{X}$ but $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}) = 0$, where $(\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}) \leftarrow \mathsf{Eval}(\mathfrak{K}, \mathcal{X})$, $\mathsf{wit}_x^{\mathcal{X}} \leftarrow \mathsf{WitCreate}(\mathfrak{K}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x)$ and $\mathsf{Proof} \leftarrow \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}_x^{\mathcal{X}}, x)$. In this case, we say that the accumulator is *correct* if the advantage of an adversary to win the game is negligible. Now, we prove the lemma by proving the contrapositive. Let $\mathcal{B}$ be an adversary that breaks the accumulator scheme correctness property with non negligible advantage. We build $\mathcal{A}$ an adversary that breaks the completeness property of the proof system. Let $\mathcal{C}$ be a challenger. $\mathcal{A}$ is given Para and λ from $\mathcal{C}$. She runs $\mathsf{Gen}(\lambda)$ and gives $\mathsf{pk}_{\mathsf{acc}}$ to $\mathcal{B}$. $\mathcal{B}$ sends $(\mathcal{X}, x)$ to $\mathcal{A}$. The latter computes $\mathsf{Sta} = (\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}) \leftarrow \mathsf{Eval}(\mathfrak{K}, \mathcal{X})$, $\mathsf{Wit} = \mathsf{wit}_x^{\mathcal{X}} \leftarrow \mathsf{WitCreate}(\mathfrak{K}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x)$ (as she knows $\mathfrak{K}$) and $\mathsf{Proof} = \mathsf{Prove}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit} = \mathsf{wit}_x)$. As $\mathcal{B}$ wins the correctness security game, we have that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}) = 0$ while $x \in \mathcal{X}$, which corresponds to $\mathsf{Verif}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Proof}) = 0$ while $(\mathsf{Para}, \mathsf{Sta}, \mathsf{Proof}) \in \mathcal{R}$. As $\mathcal{B}$ wins with non negligible advantage, then so does $\mathcal{A}$.

Lemma 2. *If the proof system* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$ *is* sound*, then the accumulator* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{CompProof}, \mathsf{Verify})$ *satisfies* collision resistance*.*

Proof. We prove the contrapositive. Let $\mathcal{B}$ be an adversary that breaks the accumulator scheme collision resistance property with non negligible advantage. We build $\mathcal{A}$ an adversary that breaks the soundness property of the proof system, using $\mathcal{B}$. Let $\mathcal{C}$ be a challenger. $\mathcal{A}$ is given Para and λ from $\mathcal{C}$. She runs $\mathsf{Gen}(\lambda)$ and gives $\mathsf{pk}_{\mathsf{acc}}$ to $\mathcal{B}$. As $\mathcal{A}$ knows $\mathsf{sk}_{\mathsf{acc}}$ she can answers to all of $\mathcal{B}$'s oracle queries. At some point, $\mathcal{B}$ sends $(\mathcal{X}, \mathsf{wit}_x, x)$ to $\mathcal{A}$. The latter computes $\mathsf{Sta} = (\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}) \leftarrow \mathsf{Eval}(\mathfrak{K}, \mathcal{X})$ and $\mathsf{Proof} = \mathsf{Prove}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit} = \mathsf{wit}_x)$. As $\mathcal{B}$ wins the collision resistance security game, we have that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}) = 1$ while $x \notin \mathcal{X}$, which corresponds to $\mathsf{Verif}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Proof}) = 1$ while $(\mathsf{Para}, \mathsf{Sta}, \mathsf{Proof}) \notin \mathcal{R}$. As $\mathcal{B}$ wins with non negligible advantage, then so does $\mathcal{A}$.

Note 2. In the rest of the paper, a witness $\mathsf{wit}_x^{\mathcal{X}}$ will be written wit_x for short, if there is no ambiguity on the associated set $\mathcal{X}$.

2.2 Overview

As mentioned earlier, accumulators have evolved over time to serve various purposes, with new properties and functionalities being added to align with these objectives. However, these additions have often been made in isolation, leading to multiple definitions of accumulators accompanied by core algorithm modifications. That makes it complicated to have an overview of accumulators and their properties. That is why in 2015, Derler *et al.* [30] proposed a unified formal model, dealing with most of existing accumulators' properties. Their work became a reference when working with accumulators. However, since 2015 new properties of accumulators have been introduced, and some functionalities, were not taken into account in the work of Derler *et al.*. In this section we present an up-to-date (informal) overview of accumulators properties following our definition for accumulators. We first list the features of accumulators, except for the *correctness* property which was already defined in Section 2.1.

Trapdoorless: an accumulator scheme is said to be *trapdoorless* if the generation algorithm Gen outputs a single public key $\mathsf{pk}_{\mathsf{acc}}$ instead of a key pair $\mathfrak{K} = (\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$. Therefore, all algorithms taking as input $\mathfrak{K}$ now take as input $\mathsf{pk}_{\mathsf{acc}}$.

Note 3. Accumulators based on collision-resistant hash functions are trapdoorless.

Evaluation: in Definition 2, the evaluation algorithm takes as input the key pair $\mathfrak{K}$. If Eval takes as input $\mathsf{sk}_{\mathsf{acc}}$ (resp. $\mathsf{pk}_{\mathsf{acc}}$) solely, we say that the accumulator has *private evaluation* (resp. *public evaluation*).

Witness Generation: regarding the way witnesses are generated in $\mathsf{WitCreate}$, the literature gives four possibilities: **i)** only using the *public key* [45], in this case the accumulator is said to have *public witness generation*, **ii)** using the *secret key* [50], in this case the accumulator has *private evaluation*, **iii)** using the public key or in a more efficient way using the secret key [4,30], **iv)** using a specially created private key, called the *evaluation key* [35].

Note 4. In the case of trapdoorless accumulator, the evaluation and the witness generation are obliviously publicly made.

Trusted, Semi-Trusted and Non-Trusted Setup [30]: the knowledge of the accumulator secret key $\mathsf{sk}_{\mathsf{acc}}$ allows an adversary to break the security of the accumulator scheme, such as the collision resistance property. Therefore a natural question arise: should we trust the third party that runs the generation algorithm Gen? Oblivously there is no need for a trusted setup in the case of trapdoorless accumulators. The question is more tricky for other accumulators, such as those based on number theoretic assumptions. Two models are defined: the *trusted* setup model in which a trusted third party runs the generation algorithm Gen and discards $\mathsf{sk}_{\mathsf{acc}}$ afterwards; the *non-trusted* model in which such trusted third party does not exist. There exists another model, proposed by Lipmaa [47]: the *semi-trusted* setup model. The idea is to divide the generation algorithm Gen into two algorithms: Gen and Setup. In this model, the adversary can control the randomness used in Setup (thus knows the secret key $\mathsf{sk}_{\mathsf{acc}}$) but she can neither

access or influence the randomness of the Gen algorithm. Notice that this model still requires a partially trusted setup, and is not generally applicable (for example it does not fit the known order group setting, refer to [30] for more details). Therefore, when considering the state of the art it seems most reasonable (regarding the efficiency of the schemes) to define a security model with respect to trusted setup as [30] did and as we will do subsequently. We emphasize that this model is compatible with all existing constructions.

Sizes requierements [21]: accumulator and witness sizes should be independent of the number of accumulated elements. More formally, for $N \in \mathbb{N}$ that represents the size of the set $\mathcal{X}$ represented by the accumulator $\mathsf{acc}_{\mathcal{X}}$, then we would like that $|\mathsf{acc}_{\mathcal{X}}| \notin O(N)$ and for any $x \in \mathcal{X}$, $|\mathsf{wit}_x| \notin O(N)$.

Boundedness [5]: an accumulator scheme $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{Verify})$ is said to be *bounded* if the generation algorithm Gen takes as additional input $\mathfrak{b} \in \mathbb{N}$, such that for all set $\mathcal{X}$ given as input of the evaluation algorithm Eval, $|\mathcal{X}| \leq \mathfrak{b}$.

Note 5. In some definitions, such as in [30], the parameter $\mathfrak{b}$ belongs to $\mathbb{N} \cup \infty$ and is always given as an input of Gen. An accumulator is then said to be *bounded* if $\mathfrak{b} \neq \infty$.

Dynamic [22]: an accumulator that additionally provides efficient algorithms (Add, Delete, WitnessUpdate) that respectively adds/removes elements from the accumulated set and the accumulator, and updates the witness accordingly. More formally, the algorithms are defined as follows:

- $\mathsf{Add}(\mathfrak{K}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, y)$: the addition algorithm takes as input the accumulator key pair $\mathfrak{K}$, an accumulator $\mathsf{acc}_{\mathcal{X}}$ for a set $\mathcal{X}$, associated auxiliary information aux and an element y to be added. If $y \in \mathcal{X}$, the algorithm returns $\perp$. Otherwise it returns the updated accumulator $\mathsf{acc}_{\mathcal{X}'}$, with $\mathcal{X}' = \mathcal{X} \cup \{y\}$, along with updated auxiliary information aux'.
- $\mathsf{Delete}(\mathfrak{K}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, y)$: the deletion algorithm takes as input the accumulator key pair $\mathfrak{K}$, an accumulator $\mathsf{acc}_{\mathcal{X}}$ for a set $\mathcal{X}$, associated auxiliary information aux and an element y to be removed. If $y \notin \mathcal{X}$, the algorithm returns $\perp$. Otherwise it returns the updated accumulator $\mathsf{acc}_{\mathcal{X}'}$, with $\mathcal{X}' = \mathcal{X} \setminus \{y\}$, along with updated auxiliary information aux'.
- $\mathsf{WitnessUpdate}(\mathfrak{K}, \mathsf{wit}_x, \mathsf{aux}, y)$: the witness update algorithm takes as input the accumulator key pair, a witness wit_x to be updated, auxiliary information aux and a value y that was added (resp. removed) to (resp. from) the accumulator, where aux indicates addition or deletion. It returns updated witness wit'_x on success, and $\perp$ otherwise.

Note 6. If the accumulator scheme only provides Add (resp. Delete) and WitnessUpdate algorithms, then we say that the scheme is *additive* (resp. *subtractive*).

Publicly Updatable [30]: a dynamic (or additive or subtractive) accumulator in which updates (of the accumulators and witnesses) are performed without the secret key.

Universal [42]: witnesses can be generated to prove membership or non-membership. The accumulator scheme now relies on two proof systems, one for proving membership and one for proving non-membership. The witness creation, the proof computation and the verification algorithms take an additional input, a boolean Type that indicates membership ($\mathsf{Type} = 0$) or non-membership ($\mathsf{Type} = 1$). More formally, an accumulator $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{CompProof}, \mathsf{Verify})$ is said to be *universal* if the syntax of the witness creation, the proof computation algorithm, and the verification algorithms are as follows: $\mathsf{WitCreate}(\mathfrak{K}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, y, \mathsf{Type})$, $\mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_x, x, \mathsf{Type})$ and $\mathsf{Verify}(\ \mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, y, \mathsf{wit}_y, \mathsf{Type})$. The witness creation algorithm outputs mwit_y for membership witness and nmwit_y for non-membership witness. When given as input $\mathsf{Type} = 0$, the algorithms $\mathsf{CompProof}$ and Verify run respectively the Prove and Verif algorithms of the membership proof system. Given as input $\mathsf{Type} = 1$, they run the non-membership proof system algorithms Prove and Verify.

Note 7. In some works, an accumulator scheme supporting only membership (resp. non-membership) proofs is said to be *positive* (resp. *negative*).

Delegatable non-membership proofs [1]: it is possible for a user to give to another entity the ability to prove non-membership of the former's element, without the latter knowing the concerned element. More formally, an accumulator with delegatable non-membership proofs has four extra algorithms $(\mathsf{Dele}, \mathsf{Vali}, \mathsf{Rede}, \mathsf{CompNMProof})$ such that Dele outputs a delegation key Del_y associated to element y; Vali verifies if a delegation key is valid; Rede computes a new delegation key from one given as input; and $\mathsf{CompNMProof}$ computes a non-membership proof from the delegation key of element y (Del_y) for an accumulator given as input. See Section 5 for more details.

Note 8. **i)** A witness for an element x is related to the accumulated set but not to the witness, whereas a delegation key is related to the x only and not the accumulated set. **ii)** Defining the delegatable property when using the [30] model is kinda complicated. The first thing to do is to separate the verification algorithm into two algorithms: the first one computes some values from the accumulator and the witness, and the second verifies if a given relation between those values is satisfied. However, defining formally for any accumulator what these values are and what is the relation to verify is not an easy task. Our definition is more suitable as it already propose two algorithms for the verification, and formally introduced and highlight the proof system used in the accumulator.

Subset query [31,35,45] and Batching [32,65]: in an accumulator scheme with *subset query*, witnesses can be generated for a subset of the accumulated set rather than individual elements. In this case, the syntax of the witness generation algorithm is the following one $\mathsf{WitCreate}(\mathfrak{K}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathcal{I})$, where $\mathcal{I} \subset \mathcal{X}$. Sometimes, direct generation is not possible thus the accumulator is using *batching* techniques [16] [2]. For example,*witness aggregation* [16] is a batching technique: first it computes individual witnesses for all elements of the subset, then aggregates the witnesses.

Multiset setting [48,35,16]: sets that can be accumulated can be multisets. Each element is associated to a count (belonging to $\mathbb{N}$), that is equal to 0 when the element is

[2] Applying a single action applied to n items instead of one action per item

not accumulated. More formally, any set $\mathcal{X}$ that is accumulated is composed of tuples of the form (x_i, k_i) for $i = 1, \cdots, |\mathcal{X}|$, where x_i is the element to accumulated and $\mathsf{k}_i \in \mathbb{N}$ represents the multiplicity of the element in the set.

Asynchronous [61]: the accumulator satisfies both **low update frequency** and **old accumulator compatibility**. An accumulator satisfies *low update frequency* if it is dynamic, and witnesses do not have to be updated at each update of the accumulator (for witnesses associated to elements not added in the accumulator). An accumulator satisfies *old accumulator compatibility* if it is dynamic, and verification still holds with an updated witness and an old (not updated) accumulator, for an element already present in the old accumulator.

Note 9. We present the asynchronous property for an additive, non-universal scheme, as done in [61]. One can easily extend this property to a dynamic universal scheme.

Accumulators and Zero-Knowledge proofs: some works (such as [22,55,4,41]) complete cryptographic accumulator with *zero-knowledge proof-of-knowledge protocols*: a client that knows his value x is (or is not) in $\mathcal{X}$, can efficiently prove to a third-party that his value is (resp. is not) in the set, without revealing x or its associated witness. Some accumulators are designed to be checked by a SNARK system efficiently, such as [23]. In this case we refer to such accumulators as *SNARKs-friendly*. Recently, Lipmaa introduced a new type of accumulator, called *determinantal* [49], that has a structure that support a special type of NIZK, called CLPØ [27].

Note 10. The formal definition of an accumulator scheme with zero-knowledge proofs, the one of a SNARK-friendly scheme, and the one of a determinantal scheme can easily be derived from our Definition 2 by replace the proof system by the appropriate NIZK. Again we here prove the modularity of our definition.

Dually computable [14]: an accumulator with two evaluation algorithms, one that takes as input only the scheme's secret key, while the other takes as input the public key solely. Outputs of both algorithms are distinguishable. More formally, an accumulator $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{CompProof}, \mathsf{Verify})$ is said to be *dually computable* if **i)** the syntax of the evaluation algorithm is $\mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathcal{X})$, **ii)** there is a second evaluation algorithm, with syntax $\mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$, **iii)** for any set $\mathcal{X}$, $\mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}}, \mathcal{X}) \neq \mathsf{PublicEval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X})$, and **iv)** the witness creation, the proof computation and the verification algorithms work with the outputs of both evaluation algorithms.

We now list all security properties found in the literature. A fundamental requirement for a secure cryptographic accumulator is collision resistance, as already presented in Section 2.1. Various definitions have been proposed in the literature, and we discuss them for both dynamic and universal accumulators. In case of static (resp. non-universal) scheme, just omit the dynamic (resp. universal) related parts. Throughout this paper, we assume adversaries are "Probabilistic Polynomial Time" (PPT).

Note 11. In Definition 4, we analyze a static non-universal accumulator scheme. In the case of a dynamic and universal accumulator, the property can be defined similarly:

the adversary gains access to an oracle for acquiring membership and non-membership witnesses, as well as an oracle for adding or deleting elements from an accumulator. The winning condition remains consistent with the previous definition: successfully meeting the condition from before or discovering an element x' in $\mathcal{X}$ to forge a non-membership witness [3].

One-Wayness [15]: informally it is hard for an adversary who is given a set $\mathcal{X} = (x_1, \cdots, x_N)$, its accumulation result $(\mathsf{acc}_{\mathcal{X}}, \mathsf{aux})$, and another value $x \notin \mathcal{X}$ (resp. $x \in \mathcal{X}$) to output a value wit such that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}, 0) = 1$, where $\mathsf{Proof} \leftarrow \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}, x, 0)$ (resp. $\mathsf{Verify}(\ \mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}, 1) = 1, \mathsf{Proof} \leftarrow \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}, x, 1)$).

Strong One-Wayness [12]: informally, given $\mathcal{X} = (x_1, \cdots, x_N)$ and $\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}$, it is hard for an adversary to output $x \notin \mathcal{X}$ (resp. $x \in \mathcal{X}$) and wit such that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}, 0) = 1$, where $\mathsf{Proof} \leftarrow \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}, x, 0)$ (resp. $\mathsf{Verify}(\ \mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}, 1) = 1$, where $\mathsf{Proof} \leftarrow \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}, x, 1)$).

Undeniability [47]: informally, it is hard for an adversary to output an accumulator acc^*, a value x and two witnesses mwit and nmwit such that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}^*, \mathsf{Proof}, 0) = 1$ and $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}^*, \mathsf{Proof}', 1) = 1)$ hold, where $\mathsf{Proof} \leftarrow \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{mwit}, x, 0)$ and $\mathsf{Proof}' \leftarrow \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{nmwit}, x, 1)$. Notice that the adversary has access to oracles $\mathcal{O}^E, \mathcal{O}^A, \mathcal{O}^D, \mathcal{O}^W$ that respectively represent the oracle for the algorithms $\mathsf{Eval}, \mathsf{Add}, \mathsf{Delete}$ and $\mathsf{WitCreate}$.

One-Way-Domain [31]: informally, the accumulator is collision resistant, and the set of values that can be accumulated is the span of a one-way function. Hence, it is computationally intractable to find witnesses for random values in the accumulator's domain. More formally, there exists a relation $\mathfrak{R}$ over $\mathcal{D}$ (the accumulator domain)$\times\mathfrak{A}$, where $\mathfrak{A}$ is another set, called the antecedent set, such that :

- (efficient verification): there exists an efficient algorithm $\mathfrak{D}$ that on input $(y, a) \in \mathcal{D} \times \mathfrak{A}$, returns 1 if and only if $(y, a) \in \mathfrak{R}$.
- (efficient sampling): there exists a probabilistic algorithm W that on input λ returns $(y, a) \in \mathcal{D} \times \mathfrak{A}$ such that $(y, a) \in \mathfrak{R}$. We refer to a as pre-image of y.
- (one-wayness): it is computationally hard to compute any pre-image a' of an element y that was sampled with W. Formally, for any PPT adversary $\mathcal{A}$: $\Pr\left[(y, a) \leftarrow W(\lambda); a' \leftarrow \mathcal{A}(\lambda, y) : (y, a') \in \mathfrak{R}\right] = \epsilon(\lambda)$, where $\epsilon(.)$ is a negligible function.

Indistinguishability [30]: informally, given the public key, the adversary chooses two sets $\mathcal{X}_0$ and $\mathcal{X}_1$ and obtain the evaluation of one of the two. It has to decide which one. Note that the adversary has access to oracles $\mathcal{O}^E, \mathcal{O}^A, \mathcal{O}^D, \mathcal{O}^W$ that represent the oracles for the algorithms Eval, Add, Delete and $\mathsf{WitCreate}$ respectively. An adversary is allowed to query them an arbitrary number of times. However, there are some restrictions regarding the oracles to prevent a trivial win by the adversary: $\mathcal{O}^A$ can only be

[3] [65] introduced the *chosen element attack* (CEA) to characterize collision resistance in dynamic accumulators. Notice that this term has been discontinued or abandoned.

ran on elements $x \notin \mathcal{X}_0 \cup \mathcal{X}_1$, $\mathcal{O}^D$ can only be ran on elements $x \in \mathcal{X}_0 \cap \mathcal{X}_1$, $\mathcal{O}^W$ when queried for $\mathsf{Type} = 0$ (*i.e.* membership) can only return witnesses for elements that belong to $\mathcal{X}_0 \cap \mathcal{X}_1$, while when queried for $\mathsf{Type} = 1$ (*i.e.* non-membership) can only return witnesses for elements that do not belong to $\mathcal{X}_0 \cup \mathcal{X}_1$.

Zero-knowledge accumulator [35]: informally, an accumulator is *zero-knowledge* if accumulated value, and (non-)membership witnesses leak nothing about the accumulated set at any given point in the security game (even after insertions and deletions, if the accumulator is dynamic).

Note 12. **i)** One requirement for zero-knowledge accumulator is to have *ephemeral* proofs, meaning that a proof generated before an update should not be valid after an update. With this condition, it is easy to see that a zero-knowledge accumulator scheme cannot be asynchronous. **ii)** In the two above definitions, the adversary is not given the auxiliary information. **iii)** Accumulators with *zero-knowledge proof-of-knowledge protocols* satisfy a privacy notion that is different from the *zero-knowledge* notion of [35] in which the entire protocol execution (as observed by a curious client or an external attacker) leaks nothing.

Obliviousness [11]: the accumulator satisfies both **element hiding** and **Add-Del indistinguishability**. An accumulator satisfies *element hiding* if publicly available auxiliary information aux output by update algorithms (Add or Delete) and associated to an accumulator does not lead any information about the elements in the accumulated set. An accumulator satisfies *Add-Del indistinguishability* if no adversary given publicly available information aux output by update algorithms (Add or Delete) can learn if an operation is an addition or a deletion. We end this section by presenting, in Table 1, a state of the art for asymmetric cryptographic accumulator schemes. The list is not exhaustive but considers all the properties seen in Section 2.

Note 13. The code-based accumulator scheme of [53] is adapted from the lattice-based accumulator scheme of [44].

3 Discussions on Accumulators Security

This section presents several discussions on the security properties of accumulators. First, we delve into the undeniability security property within the trusted model setup. Following that, we discuss the obliviousness property. Finally, we summarize existing relations between these properties and introduce new connections. It is worth noting that we did not specifically address the relationships between certain properties of accumulators and functionalities, as this aspect has already been explored in existing works [8].

[4] The size in constant only in some cases, otherwise it is strictly less than logarithmic in the number of values accumulated.

[5] This scheme is working can be working with an additional input of the trapdoor $\tilde{\mathsf{sk}}_{\mathsf{acc}}$. Without it, the updates are possible only by computing the new accumulator from scratches. Thus the scheme is not really publicly updatable.

Table 1. Asymmetric accumulators comparison. A • means that the functionality/property is satisfied. A $\approx$ means that the scheme satisfies a property slightly different. A $\Rightarrow$ means that the scheme satisfies a property as the latter is implied by a property satisfied by the scheme. A $\oplus$ means that the scheme is additive. "T", "NT" and "ST" respectively mean "trusted", "not trusted" and "semi trusted". "ZK" and "Determ" respectively means "zero-knowledge" and "determinantal". "EK" in the witness generaiton column means that

		Functionalities						Properties											Security							
Type	Schemes	Trapdoorless	Evaluation	Witness Generation	Setup	Constant size acc. value	Constant size witness	Boundedness	Dynamic	Publicly Updatable	Universal	Delegatable NM proofs	Subset query	Batch updates	Multiset	Asynchronous	ZK Proofs of Knowledge	Dually Computable	One-way	Strong one-way	Collision resistant	Undeniable	One-way domain	Indistinguishable	Zero-knowledge	Obliviousness
RSA -Based	[15]		Public	Public	T	•	•												•							
	[12]		Public	Public	T	•	•												⇒	⇒	•					
	[62]	•	Public	Public	NT	•	•														≈					
	[22]		Public	Public	T	•	•		•	•							ZK		⇒	⇒	•					
	[31]		Public	Public	T	•	•		•	•									⇒	⇒	•		•			
	[42]		Public	Public	T	•	•		•	•	•								⇒	⇒	•					
	[47]		Public	Public	ST	•	•				•								⇒	⇒	⇒	•				
	[16]	•	Public	Public	NT	•	•		•	•	•			•					⇒	⇒	⇒	•				
	[23]	•	Public	Public	NT	•	•		•	•	•			•			SNARK		⇒	⇒	⇒	•				
	[65]		Private	Private	T														⇒	⇒	•					
	[10]	•	Public	Public	NT				•	•	•								⇒	⇒	⇒	•				•
Hash- Based	[18]	•	Public	Public	NT				•	•	•								⇒	⇒	⇒	•				
	[20]		Public	Public	NT	•			•	•	•											≈				
	[61]	•	Public	Public	NT				⊕	•						•			⇒	⇒	•					
	[17]	•	Public	Public	NT												ZK		⇒	⇒	•					
	[38]		Private	Private	T		≈[4]				•								⇒	⇒	⇒	•				
Lattice -Based	[57]	•	Public	Public	NT				•										⇒	⇒	•					
	[37]		Public	Private	T	•	•												⇒	⇒	•					
	[44]	•	Public	Public	NT	•											ZK		⇒	⇒	•					
	[46]	•	Public	Public	NT	•			•	•							ZK		⇒	⇒	•					
	[67]	•	Public	Public	NT	•					•						ZK		⇒	⇒	•					
	[68]		Public	Secret	T	•	•	•	•		•						ZK		⇒	⇒	⇒	•		•		
Code -based	[53]	•	Public	Public	NT												ZK		⇒	⇒	•					
	[6]	•	Public	Public															⇒	⇒	•			•		
Pairing- Based	[55]		Public	Public	T	•	•		•	•									⇒	⇒	•					
	[5]		Public	Public	T	•	•	•	•	•									⇒	⇒	•					
	[29]		Public	Public	T	•	•	•	•	•	•								⇒	⇒	•					
	[21]		Private	Private	T	•	•												⇒	⇒	•					
	[4]		Public	Both	T	•	•	•	•	•	•								⇒	⇒	•					
	[2]		Public	Public	T	•	•		•	•	•	•							⇒	⇒	•					
	[30]		Both	Both	T	•	•	•	•	≈[5]									⇒	⇒	•			•		
	[35]		Private	EK	T	•	•	•	•		•								⇒	⇒	•			⇒	•	
	[45] s.1		Public	Public	T	•	•	•					•						⇒	⇒	•					
	[45] s.2		Public	Public	T	•	•	•			•								⇒	⇒	⇒	•		•		
	[49]		Public	Public	T	•	•	•			•						Determ		⇒	⇒	•					
	[14]		Both	Public	T	•	•	•					•		•			•	⇒	⇒	•					

3.1 Discussion about Undeniability in the Trusted Setup Model

According to [35], in the trusted setup model undeniability provides more than what is necessary in terms of security. We formalize this statement, and we prove it.

Theorem 2. *In the trapdoor setting, trusted model setup, if the evaluation is done* privately*, then the undeniability property is an overkill; the collision resistance property is enough.*

Proof. In the undeniability security game, when the only way for the adversary $\mathcal{A}$ to compute acc^* is to request the challenger (private evaluation) by giving a set $\mathcal{X}^*$ we need to consider both cases:

- If $x^* \in \mathcal{X}^*$, then $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{Proof}, 0) = 1$, by definition, where $\mathsf{Proof} = \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{aux}, \mathsf{mwit}_{x^*}, x^*, 0)$, $\mathsf{mwit}_{x^*} \leftarrow \mathsf{WitCreate}(\mathfrak{K}, \mathcal{X}^*, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{aux}, x^*, 0)$. To win the game, $\mathcal{A}$ must find a non-membership witness nmwit'_{x^*} such that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{Proof}', 1) = 1$, where $\mathsf{Proof}' = \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{aux}, \mathsf{nmwit}'_{x^*}, x^*, 1)$. This means that $\mathcal{A}$ wins the undeniability game if it wins the collision resistant game.
- If $x^* \notin \mathcal{X}^*$, $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{Proof}, 1) = 1$, by definition, where $\mathsf{Proof} = \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{aux}, \mathsf{nmwit}_{x^*}, x^*, 1)$, $\mathsf{nmwit}_{x^*} \leftarrow \mathsf{WitCreate}(\mathfrak{K}, \mathcal{X}^*, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{aux}, x', 1)$. To win the game, $\mathcal{A}$ must find a membership witness mwit'_{x^*} such that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{Proof}', 1) = 1$, where $\mathsf{Proof}' = \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}^*}, \mathsf{aux}, \mathsf{mwit}'_{x^*}, x^*, 0)$. This means that $\mathcal{A}$ wins the undeniability game if it wins the collision resistant game.

  In both cases, collision-resistance is enough, and then undeniability is not required.

Note 14. If the evaluation is done publicly, *i.e.*, without the knowledge of $\mathsf{sk}_{\mathsf{acc}}$, then undeniability is required. Also (and obliviously) in the non-trusted setup model this property is also required.

3.2 Discussion on Obliviousness

Before the recent work of Baldimtsi *et al.* [11], only two privacy-preserving properties existed for accumulators: *indistinguishability* and *zero-knowledge*. Both demand that the adversary lacks access to the accumulator's auxiliary information aux, crucial for preventing leaks about the accumulated set. This precaution is essential because auxiliary information could potentially disclose details about the accumulated set, particularly when the latter is included in the auxiliary information or when the auxiliary information after an update reveals the added/removed element. Obliviousness [11] goes further, ensuring that publicly available information doesn't disclose anything about the set, including its size. This property focuses on enhancing privacy during update algorithm execution (*i.e.,* aux'). The authors introduced some secret information during the Add algorithm to hide the added element, which is also used in witness generation and verification. However, in accumulator schemes, verification should rely on public elements alone, making it challenging for an oblivious accumulator scheme. While acknowledging the importance of protecting information leaked by aux, we remain unconvinced that the proposed solution effectively addresses this concern.

3.3 Relations Between Security Properties

Looking at accumulators' security properties, we classify them into two categories: those that protect the witness (i.e., preventing forgery of witnesses), and those that protect the accumulated set (i.e., hiding information about the set). In the first category, we have: (strong) one-wayness, collision resistance, one-way domain, and undeniability. In the second category, we have: indistinguishability, zero-knowledge, and obliviousness. It's worth noting that the properties in the first category are computational, while in the second category, they are decisional. Also, there is no security property that protects the accumulated value, perhaps because the latter is mostly computed publicly.

Note 15. Properties that protect the accumulated set define privacy security for accumulators schemes. As already observed in [51,52,30], when formulating a notion of privacy for cryptographic accumulators the fact that the accumulation value computation must be randomized becomes evident.

Comparison between properties of the second category. The notion of zero-knowledge differs from the privacy notion *indistinguishability* of [30], by protecting not only the originally accumulated set but also all subsequent updates. In fact, [35] formally proved in Section 3.3 the following theorem that states that for cryptographic accumulators, zero-knowledge is a strictly stronger property than indistinguishability.

Theorem 3. *Every zero-knowledge dynamic universal accumulator is indistinguishable under the definition of [30], while the opposite is not always true.*

While being really similar at first glance, zero-knowledge and obliviousness are actually different: in the former auxiliary information aux is not given while it is in the latter. It seems then that obliviousness is stronger than zero-knowledge. However, obliviousness requires some particular requirements in the accumulator's algorithms. Thus it cannot be applied to all schemes. Plus taking into account the above discussion, we decided not to include this property in our comparison.

Relations between other properties. First, as the adversary is given more and more flexibility, it is easy to see that the theorem below holds, while the opposite is not true.

Theorem 4. *Every accumulator satisfying strong one-wayness satisfies one-wayness; every collision resistant accumulator satisfies strong one-wayness; every one-way domain accumulator is collision resistant.*

Due to lack of space, we furnish the proof that every collision resistance accumulator satisfies strong one-wayness. The rest of the proof can easily be derived.

Proof. We prove the contrapositive: we suppose that there exists an adversary $\mathcal{B}$ that breaks the strong one-wayness property, and we build an adversary $\mathcal{A}$ that breaks the collision resistance property, using $\mathcal{B}$. Let $\mathcal{C}$ be the challenger of the collision resistance security game. $\mathcal{C}$ run $\mathsf{Gen}(\lambda)$ to get $\mathfrak{K} = (\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$ and sends $\mathsf{pk}_{\mathsf{acc}}$ to $\mathcal{A}$, who sends it to $\mathcal{B}$. $\mathcal{A}$ then chooses a set $\mathcal{X}$ and queries the oracle $\mathcal{O}^E$ to get $\mathsf{acc}_{\mathcal{X}}$. Then, she sends $\mathcal{X}, \mathsf{acc}_{\mathcal{X}}$ to $\mathcal{B}$. The latter returns an element $x^{'} \notin \mathcal{X}$ and a membership witness $\mathsf{wit}^{'}$ such

that $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}, 0) = 1$, where $\mathsf{Proof} = \mathsf{CompProof}(\mathfrak{K}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}^{'}, x^{'}, 0) = 1$ with non negligible probability. Therefore, $\mathcal{A}$ outputs $(\mathcal{X}, x^{'}, \mathsf{wit}^{'})$ and wins the collision resistance security game with non negligible advantage.

For undeniability, the following lemma has been proven in Appendix C.1 of [30]

Lemma 3. *Every undeniable universal accumulator is collision-resistant.*

As mentioned in [47], a black-box reduction in the other direction is impossible. In particular, [19] provides a collision-resistant universal accumulator and exhibit an example to show that their scheme is not undeniable. This proves the following lemma.

Lemma 4. *Not every collision resistant scheme is undeniable.*

It remains to make the link between undeniability and one-way domain. At first, we focus on the scheme based on sorted hash tree given in [19]. This one is proven to be universal and collision resistant, and as state before it is not undeniable. It can moreover be used for domain that is in the span of a one-way function. Hence, one-way domain does not imply undeniability. Therefore, we can establish the following lemma that is proven using the above counterexample.

Lemma 5. *Not every one-way domain accumulator is undeniable.*

For the opposite, we do not succeed in proving that this is true or false, and we leave it as an open problem. In Figure 1, we summarize all the above properties and their relation, based on related work, but also on our new results. In the figure an arrow means "implies", a crossed out arrow means "does not imply" and a dash arrow means "not proven". Notice that as there is no relation between *obliviousness* and other properties we do not include the former in the figure.

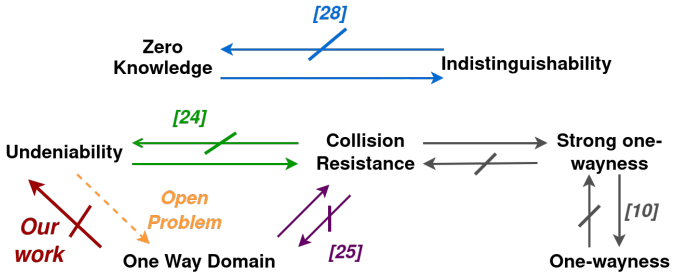

Fig. 1. Relations between security properties of accumulators.

4 New Security Property

In a centralized cryptocurrency system, the accumulator represents spent transaction outputs, enabling users to verify specific transactions through a publicly generated membership witness. Until recently, there was no accumulator scheme offering both

private evaluation and public evaluation. Consequently, the scheme depended on a signature scheme to ensure the accuracy of the accumulator in representing approved transactions. Bridging this gap, Barthoulot *et al.* [14] introduced the first accumulator scheme with private evaluation and public witness generation. However, utilizing their scheme in the described scenario lacks a mechanism (distinct from the signature) to safeguard the accumulator. Our contribution addresses this security concern by introducing a novel property, making it challenging to "forge" a privately computed accumulator that passes verification with a legitimate witness. Implementing an accumulator with this property eliminates the need for a signature, simplifying the overall system.

Definition 5. ***Unforgeability of private evaluation (UPE).*** *A static non-universal accumulator scheme with private evaluation and public generation is said to satisfy* unforgeability of private evaluation *if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(.)$ such that, for any y chosen randomly in $\mathcal{X}^*$:*

$$\Pr\left[\begin{array}{c}(\mathsf{sk}_{\mathsf{acc}},\mathsf{pk}_{\mathsf{acc}})\leftarrow\mathsf{Gen}(\lambda),(\mathcal{X}^*,\mathsf{acc}^*)\leftarrow\mathcal{A}(\mathsf{pk}_{\mathsf{acc}});\\ (\mathsf{acc}_{\mathcal{X}^*},\mathsf{aux})\leftarrow\mathsf{Eval}(\mathsf{sk}_{\mathsf{acc}},\mathcal{X}^*),y\leftarrow\mathcal{X}^*;\\ \mathsf{wit}_y\leftarrow\mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}},\mathcal{X}^*,\mathsf{acc}_{\mathcal{X}^*},\mathsf{aux},y)\\ \mathsf{Proof}\leftarrow\mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}},\mathsf{acc}_{\mathcal{X}^*},\mathsf{aux},\mathsf{wit}_y,y):\\ \mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}},\mathsf{acc}^*,\mathsf{Proof})=1\end{array}\right]\leq\epsilon(\lambda),$$

In other words, the adversary cannot convincingly demonstrate that they honestly computed an accumulator for the chosen set $\mathcal{X}^*$. Therefore, the proof must be rejected for any $y \in \mathcal{X}^*$ (except with negligible probability), which is why y is randomly selected in the definition.

Note 16. This definition, which aims to address a gap in accumulator security, might also be useful for advancing a study presented at CFail 2023 [13] where the authors attempt to establish a connection between a primitive known as *locally verifiable aggregate signatures* and asymmetric accumulators. The authors fail to prove this connection, partly due to the absence of a security property for an accumulator that can be considered analogous to the unforgeability of signature schemes.

In the following we prove that [14] accumulator satisfies our new security property. Before to present [14]'s scheme, we recall informally some notation and definition. First, for any group element g and any vectors $\boldsymbol{v} = (v_1, \cdots, v_l)$, $\boldsymbol{u} = (u_1, \cdots, u_l)$, we denote by $g^{\boldsymbol{v}}$ the vector $(g^{v_1}, \cdots, g^{v_l})$ and define $e(g^{\boldsymbol{v}}, g^{\boldsymbol{u}}) := \prod_{i=1}^{l} e(\, g^{v_i}, g^{u_i}) = e(g,g)^{\boldsymbol{v}\cdot\boldsymbol{u}}$. Let $\mathbb{B} = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n)$ and $\mathbb{B}^* = (\boldsymbol{b}_1^*, \cdots, \boldsymbol{b}_n^*)$ be two basis of $\mathbb{Z}_p^n$ (p prime, n fixed dimension). The two basis are **dual orthonormal**, meaning that $\boldsymbol{b}_i \cdot \boldsymbol{b}_j^* = 0 \pmod p$ whenever $i \neq j$, and $\boldsymbol{b}_i \cdot \boldsymbol{b}_i^* = \psi \pmod p$ for all i, where ψ is a uniformly random element of $\mathbb{Z}_p^*$. A tuple $(\mathbb{B}, \mathbb{B}, \psi)$, called **Dual pairing vector spaces** (DPVS) [56,26], is generated by the algorithm $\mathsf{Dual}(\mathbb{Z}_p^n)$. We now briefly present [14]'s scheme, which is bounded by $q \in \mathbb{N}$:

- The secret key is $\mathsf{sk}_{\mathsf{acc}} = (s, \mathbb{D}, \mathbb{D}^*)$, where $(\mathbb{D}, \mathbb{D}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^2)$, $\psi \in \mathbb{Z}_p$ is the random such that $\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^* = \boldsymbol{d}_2 \cdot \boldsymbol{d}_2^* = \psi$, and s is a random element of $\mathbb{Z}_p^*$. The public key is $\mathsf{pk}_{\mathsf{acc}} = \left(\Gamma, g_1^{\boldsymbol{d}_2}, g_1^{\boldsymbol{d}_2 s}, \cdots, g_1^{\boldsymbol{d}_2 s^q}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_2^*}, \;\; g_2^{\boldsymbol{d}_2^* s}, \cdots, g_2^{\boldsymbol{d}_2^* s^q}\right)$, where $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, \frac{g}{2})$ is an asymmetric bilinear group.

– For a set $\mathcal{X}$, its accumulator is $\mathsf{acc}_{\mathcal{X}} = g_1^{\boldsymbol{d}_1 \sum_{i=0}^{q} a_i s^i} \in \mathbb{G}_1^2$, where $\{a_i\}_{i=0,\cdots,q}$ are the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X}}[Z] = \prod_{x\in\mathcal{X}}(Z+x)$.
– For an element y, its witness is $\mathsf{wit}_y = g_2^{\boldsymbol{d}_2^* \sum_{i=0}^{q} b_i s^i}$, where $\{b_i\}_{i=0,\cdots,q}$ are the coefficients of the polynomial $\mathsf{Ch}_{\mathcal{X}\setminus\{y\}}[Z] = \prod_{x\in\mathcal{X}\setminus\{y\}}(x+Z)$.
– The verification is done by checking if $e(\mathsf{acc}_{\mathcal{X}}, g_2^{\boldsymbol{d}_1^*}) = e(g_1^{\boldsymbol{d}_2(y+s)}, \mathsf{wit}_y)$.

To prove that the scheme satisfies unforgeability of private evaluation, we introduce the following assumption, that can be reduced to CDH, as we prove in Appendix A. This assumption is the first *computational* assumption for dual pairing vector spaces, and therefore might be of independent interest for future works.

Definition 6. ***Fixed argument dual pairing vector spaces inversion assumption****(FA-DPVS-I). Let $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be an asymmetric bilinear pairing group and $(\mathbb{D}^*, \mathbb{D}) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^2)$ be two dual orthonormal bases. The assumption states that given $(\Gamma, g_1^{\boldsymbol{d}_2}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_2^*})$ it is hard to compute $g_1^{\boldsymbol{d}_1}$.*

Theorem 5. *If the fixed argument dual pairing vector spaces inversion assumption holds, then [14]'s accumulator satisfies unforgeability of private evaluation.*

Proof. We prove the contrapositive. Let $\mathcal{B}$ be an adversary that breaks [14]'s scheme UPE security with non negligible advantage. We build $\mathcal{A}$ an adversary that uses $\mathcal{B}$ to break FA-DPVS-I assumption. $\mathcal{A}$ is given $(\Gamma, g_1^{\boldsymbol{d}_2}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_2^*})$. She chooses $s \leftarrow \mathbb{Z}_p$, creates $\mathsf{pk}_{\mathsf{acc}}$ and sends it to $\mathcal{B}$. $\mathcal{B}$ answer to $\mathcal{A}$ with a tuple of message-forged accumulator $(\mathcal{X}^*, \mathsf{acc}^*)$. $\mathcal{A}$ knows that for any $y \in \mathcal{X}^*$, $e(\mathsf{acc}^*, g_2^{\boldsymbol{d}_1^*}) = e(g_1^{\boldsymbol{d}_2(y+s)}, \mathsf{wit}_y)$ and that $e(g_1^{\boldsymbol{d}_2(y+s)}, \mathsf{wit}_y) = e(g_1, g_2)^{\psi \sum_{i=1}^{q} a_i s^i}$. Thus $e(\mathsf{acc}^*, g_2^{\boldsymbol{d}_1^*}) = e(g_1, g_2)^{\psi \sum_{i=1}^{q} a_i s^i}$. Thanks to the knowledge of $\mathcal{X}^*$ and s, $\mathcal{A}$ can recover $\{a_i\}_{i=0}^{q}$, computes $(\sum_{i=0}^{q} a_i s^i)^{-1}$ and obtains that $e((\mathsf{acc}^*)^{(\sum_{i=0}^{q} a_i s^i)^{-1}}, g_2^{\boldsymbol{d}_1^*}) = e(g_1, g_2)^{\psi}$. $\mathcal{A}$ outputs $(\mathsf{acc}^*)^{(\sum_{i=0}^{q} a_i s^i)^{-1}}$ as her answer and wins the game with an advantage equal to $\mathcal{B}$'s advantage, therefore with non-negligible advantage.

5 Delegatable Proofs

In this section we focus on a property introduced in 2010 by Acar and Nguyen [2]: *delegatable* non-membership proofs. Our aim is to understand what is necessary to obtain delegatable proofs. Before to do this, we briefly recall some applications of accumulators to highlight the interest of the delegatable property.

Accumulators' applications. As already mentioned in the introduction, originally accumulators served purposes such as timestamping and membership testing [15]. Their applications expanded to include fail-stop signatures [12], membership revocation in group signatures [22], anonymous credentials (delegatable)[2], and e-cash [5] along others. This list is not exhaustive; detailed applications are covered in surveys such as [59]. An intriguing observation is that, while cryptographic accumulators aim to maintain the size of cryptographic objects as constant, they are infrequently incorporated into encryption schemes. Works like [3,34,66] explore this avenue. [34,3] propose

broadcast encryption schemes using cryptographic accumulators (based on RSA) for managing users' secret keys. Wang and Chow [66] introduce an identity-based broadcast encryption scheme relying on a simplified form of accumulators. They leverage the compactness of accumulator outputs for scheme efficiency but do not consider other accumulator functionalities. Some research incorporates accumulators to add revocation functionality to existing encryption schemes. For instance, [39] adds revocation to Lewko and Waters' hierarchical identity-based encryption scheme [40]. Notably, [14] proposes a scheme using cryptographic accumulators for both key management and encryption, making it the only known scheme utilizing accumulators for encryption. They employ dually computable accumulators to construct attribute-based encryption schemes, albeit with a larger public key size, paving the way for future works in building encryption schemes from accumulators.

Applications of delegatable proofs. Delegatable (non-)membership proofs, initially designed for anonymous credentials, find utility in access control systems and permission delegation in distributed environments. As accumulators gain traction in encryption schemes, a promising avenue involves crafting a re-encryption proxy from an accumulator. By integrating delegation into an accumulator-based encryption scheme, the potential for establishing a re-encryption proxy arises. Also, accumulators hold significance in blockchain and digital cash. Introducing delegatable proofs can significantly enhance the efficiency of both systems while preserving privacy. Our goal is to discover a generic method for obtaining accumulators with this property. Focusing solely on delegatable non-membership proofs, we simplify the discussion for clarity, noting that the insights presented apply equally to delegatable membership proofs. We begin by formally define an accumulator scheme with delegatable non-membership proofs.

Definition 7. ***Delegatable non-membership proofs[2].*** *A universal accumulator* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{CompProof}, \mathsf{Verify})$ *allows delegatable non-membership proofs if it additionally provides the following algorithms.*

- $\mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y)$*: the delegation algorithm takes as input the public key* $\mathsf{pk}_{\mathsf{acc}}$ *and an element* y*. It outputs a delegating key* Del_y*.*
- $\mathsf{Vali}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y)$*: the validation algorithm takes as input the public key* $\mathsf{pk}_{\mathsf{acc}}$ *and a delegating key* Del_y*. If* Del_y *is valid it returns* 1*, otherwise it returns 0.*
- $\mathsf{Rede}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y)$*: the re-delegation algorithm takes as input the public key* $\mathsf{pk}_{\mathsf{acc}}$ *and a delegating key* Del_y*. If* $\mathsf{Vali}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y) = 1$*, the algorithm returns an other delegating key* $\mathsf{Del}_x^{'}$*, otherwise it outputs* $\perp$*.*
- $\mathsf{CompNMProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y, \mathcal{X}, \mathsf{acc}_{\mathcal{X}})$*: the proof computation algorithm takes as input the public key* $\mathsf{pk}_{\mathsf{acc}}$*, a delegating key* Del_y*, a set* $\mathcal{X}$ *and the associated accumulated value* $\mathsf{acc}_{\mathcal{X}}$*. It returns a non-membership proof.*

These algorithms verify, for every PPT adversaries $\mathcal{A}, \mathcal{A}_1, \mathcal{A}_2$:

– *Delegability*: it states that a proof computed using a delegation key is indistinguishable from a proof computed using a witness if the following is negligible

$$\left| \Pr \left[\begin{array}{c} \mathfrak{K} = (\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Gen}(\lambda); (y, \mathcal{X}) \leftarrow \mathcal{A}_1(\mathsf{pk}_{\mathsf{acc}}); \\ (\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}) \leftarrow \mathsf{Eval}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}); \\ \mathsf{wit}_y \leftarrow \mathsf{WitCreate}(\mathsf{pk}_{\mathsf{acc}}, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, y, \mathsf{Type} = 1); \\ \mathsf{Proof}_0 \leftarrow \mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, \mathsf{wit}_y, y, \mathsf{Type} = 1); \\ \mathsf{Del}_y \leftarrow \mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y); \\ \mathsf{Proof}_1 \leftarrow \mathsf{CompNMProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y, \mathcal{X}, \mathsf{acc}_{\mathcal{X}}); \\ b \leftarrow \{0,1\}; b^{'} \leftarrow \mathcal{A}_2(\mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_y, \mathsf{Del}_y, \mathsf{Proof}_b) : b = b^{'} \end{array} \right] - \frac{1}{2} \right|$$

– *Unlinkability*: this property states that a delegation key for y_0 is indistinguishable from a delegation key for y_1 if the following is negligible

$$\left| \Pr \left[\begin{array}{c} \mathfrak{K} = (\mathsf{sk}_{acc}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Gen}(\lambda); (y_0, y_1) \leftarrow \mathcal{D}; \mathsf{Del}_y \leftarrow \mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y_0); \\ b \leftarrow \{0,1\}; \mathsf{Del}_{y_b} \leftarrow \mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y_b); b^{'} \leftarrow \mathcal{A}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y, \mathsf{Del}_{y_b}) : b = b^{'} \end{array} \right] - \frac{1}{2} \right|$$

– *Redelegability*: this property states that a delegation key output by the algorithm Rede is indistinguishable from a delegation key output by the algorithm Dele if the following is negligible

$$\left| \Pr \left[\begin{array}{c} \mathfrak{K} = (\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Gen}(\lambda); y \leftarrow \mathcal{A}_1(\mathsf{pk}_{\mathsf{acc}}); \mathsf{Del}_y \leftarrow \mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y); \\ \mathsf{Del}_y^0 \leftarrow \mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y); \mathsf{Del}_y^1 \leftarrow \mathsf{Rede}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y); b \leftarrow \{0,1\}; \\ b^{'} \leftarrow \mathcal{A}_2(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y, \mathsf{Del}_y^b) : b = b^{'} \end{array} \right] - \frac{1}{2} \right|$$

– *Verifiability*: this property states that a delegation key generated honestly will always pass the Vali algorithm while this is not the case for a not honestly computed delegation key, if the following are negligible

$$\left| \Pr \left[\begin{array}{c} \mathfrak{K} = (\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Gen}(\lambda); x \leftarrow \mathcal{A}(\mathsf{pk}_{\mathsf{acc}}); \mathsf{Del}_y \leftarrow \mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y) : \\ \mathsf{Vali}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_x) = 1 \text{ if } y \in \mathcal{D} \end{array} \right] - 1 \right|$$

$$\left| \Pr \left[\begin{array}{c} (\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Gen}(\lambda); \mathsf{Del}^{'} \leftarrow \mathcal{A}(\mathsf{pk}_{\mathsf{acc}}) : \mathsf{Vali}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}^{'}) = 0 \\ \text{if } \mathsf{Del}^{'} \notin \left\{ \mathsf{Del} \,\middle|\, \mathsf{Del} \leftarrow \mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y^{'}); y^{'} \in \mathcal{D} \right\} \end{array} \right] - 1 \right|,$$

where the condition $\mathsf{Del}^{'} \notin \left\{ \mathsf{Del} | \mathsf{Del} \leftarrow \mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y^{'}); y^{'} \in \mathcal{D} \right\}$ means that the delegation key $\mathsf{Del}^{'}$ does not correspond to a delegation key correctly computed, for any element $y^{'}$ of the domain $\mathcal{D}$.

How to obtain delegatable proofs? To the best of our knowledge, only one accumulator provides delegatable non-membership proofs: [2]. The key idea proposed by Acar and Nguyen is to use a specific type of proof system: one that has *homomorphic proofs*, a concept that they introduced. Informally, a proof system is said to be homomorphic if it is associated with a law, denoted $+_{\Pi}$, such that the result of $+_{\Pi}((\mathsf{Sta}_1, \mathsf{Wit}_1, \mathsf{Proof}_1), (\mathsf{Sta}_1, \mathsf{Wit}_2, \mathsf{Proof}_2))$, denoted $(\mathsf{Sta}, \mathsf{Wit}, \mathsf{Proof})$, is a valid tuple composed of a proof Proof computed from the statement Sta and witness Wit. Therefore, if the accumulator

can be expressed as a linear combination of public elements, then a delegating key will correspond to a set of proofs (one per public element). Constructing the proof associated with the statement corresponding to the accumulator is done by computing the correct operation on the proofs.

Efficiency and Aggregation (Batching). As described, a delegation key is a set of proofs. Therefore its size is dependent on the number of basis elements in the public key, which might be high. A solution to improve the efficiency of the scheme is to use a proof system with *aggregation* techniques (as done by Acar and Nguyen): the delegation key is not a set of proofs, but an aggregation (or any batch) of the proofs, *i.e.* one proof. In the following, we suppose that our non-membership proof system also has batching techniques, represented by the algorithm Batch that batches proofs, an algorithm $\mathsf{BatchVerif}$ that verifies a batched proof, knowing the associated set of statements, and a extracting algorithm $\mathsf{Extract}$ that extract from the batched proof all the proofs. Let us now see formally that all properties of a delegatable accumulator can be achieved using the underlying proof system properties.

Delegatable non-membership proofs and proof systems. First, we define the proof system properties that we will need: *witness indistinguishability* and *randomizable*.

Definition 8. ***Witness indistinguishability[36,1].*** *A proof system is said to satisfy* witness indistinguishability *if for any malicious verifier $\mathcal{V}$, the following is negligible:*

$$\Pr\left[\begin{array}{l}\mathsf{Para} \leftarrow \mathsf{Setup}(\lambda, \mathcal{R}), (\mathsf{Sta}, \mathsf{Wit}_0, \mathsf{Wit}_1) \leftarrow \mathcal{V}(\mathsf{Para}), b \leftarrow \{0,1\}, \\ \quad \mathsf{Proof}_b \leftarrow \mathsf{Prove}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit}_b), b' \leftarrow \mathcal{V}(\mathsf{Proof}_b) : b' = b\end{array}\right].$$

Definition 9. ***Randomizable proof system [1].*** *A proof system is said to be* randomizable *if has another PPT algorithm* $\mathsf{RandProof}$ *that takes as input a tuple* $(\mathsf{Para}, \mathsf{Sta}, \mathsf{Proof})$ *of setup parameters* Para*, statement* Sta *and proof* Proof *and returns another valid proof* Proof'*, which is indistinguishable from a proof produced by* Prove*.*

Let us rewrite the additional algorithms $\mathsf{Dele}, \mathsf{Rede}, \mathsf{Vali}, \mathsf{CompProof}$ required to obtain an accumulator with delegatable non-membership proofs to highlight the non-membership proof system $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$. Doing so, we can see that the properties of *witness indistinguishability* and *randomizable* of the proof system guarantees *unlinkability* and *redelegability*, while *Verifiability* comes directly from the proof system completeness and soundness. Notice that the proof system parameters Para are included in $\mathsf{pk}_{\mathsf{acc}}$, and that there exists an algorithm $\mathsf{CompWit}$ that takes as input public parameters Para, statement Sta and an element y, and returns a witness Wit_y for y.

- $\mathsf{Dele}(\mathsf{pk}_{\mathsf{acc}}, y)$: the algorithm extracts the public parameters Para from the accumulator public key $\mathsf{pk}_{\mathsf{acc}}$, and from Para it extracts the basis elements that forms a set of statement $\{\mathsf{Sta}_l\}_l$. For each l it runs $\mathsf{CompWit}(\mathsf{Para}, \mathsf{Sta}_l, y)$ to get Wit_l and then computes $\mathsf{Prove}(\mathsf{Para}, \mathsf{Sta}_l, \mathsf{Wit}_l)$ to get Proof_l. It runs $\mathsf{Batch}(\mathsf{Para}, \{\mathsf{Sta}_l, \mathsf{Proof}_l\}_l)$ to get Proof and outputs $\mathsf{Del}_y = \mathsf{Proof}$.
- $\mathsf{Vali}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y)$: the algorithms runs the batch verification algorithm $\mathsf{BatchVerif}$ on public parameters Para (included in $\mathsf{pk}_{\mathsf{acc}}$), statement $\{\mathsf{Sta}_l\}_l$ and the batched proof Del_y.

- $\mathsf{Rede}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y)$: if $\mathsf{Vali}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y) = 1$, the algorithm runs the randomization algorithm $\mathsf{RandProof}$ on public parameters Para, statement $\{\mathsf{Sta}_l\}$ and proof Proof to get a randomized proof Proof'.
- $\mathsf{CompProof}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{Del}_y, \mathcal{X}, \mathsf{acc}_{\mathcal{X}})$: from Para ko;i, $\mathcal{X}$ and $\mathsf{acc}_{\mathcal{X}}$ the algorithms finds the linear relation between $\mathsf{acc}_{\mathcal{X}}$ and the basis elements contained in Para. Then, it first extracts the proofs Proof_l of Del_y, then it uses homomorphic property of the proof system to obtain Proof_y, and finally it uses the randomization algorithm $\mathsf{RandProof}(\mathsf{Para}, \mathsf{acc}_{\mathcal{X}}, \mathsf{Proof}_y)$ to get a randomized proof Proof'_y that it outputs.

Lemma 6. *Verifiability is satisfied thanks to the completeness and soundness of the non-membership proof system.*

Proof. First, let us see that the first condition is satisfied if the proof system scheme satisfies completness. Let $y \in \mathcal{D}$ and $\{\mathsf{Sta}_l\}_l$ be the basis elements. Then Dele computes honestly $\left\{\mathsf{Wit}_y^l\right\}_l$, from $\mathsf{CompWit}$, $\{\mathsf{Proof}_l\}_l$, and Proof from Batch. Then, from completeness, we have that the probability that $\mathsf{BatchVerif}$ returns 1 is equal to 1. Thus, as Vali runs $\mathsf{BatchVerif}$, we have the first condition. Then, it is easy to see that if the second condition does not hold, that means that the underlying proof system does not satisfy soundness. Indeed, if there is an adversary that can creates a fake delegation key that passes the verification algorithm, we can create an adversary to win the soundness game, using the adversary against verifiability's second condition.

Lemma 7. *Redelagability is satisfied thanks to the randomizable property of the non-membership proof system.*

Proof. Let us see that if there is an adversary, let us say $\mathcal{B}$, that breaks the redelagability property, then we can build an adversary, denoted $\mathcal{A}$, that breaks the randomizable property of the proof system. First, $\mathcal{A}$ is given Para from the challenger, and she simulates the accumulator challenger by computing $\mathsf{pk}_{\mathsf{acc}}$, that she sends to $\mathcal{B}$. The latter chooses y that she sends to $\mathcal{A}$. $\mathcal{A}$ then creates the witnesses $\left\{\mathsf{Wit}_y^l\right\}_l$ that she sends to the challenger, along with $\{\mathsf{Sta}_l\}_l$. The challenger computes $\{\mathsf{Proof}_l\}$, then runs Batch to get Proof. She picks $b \in \{0, 1\}$: if $b = 0$, she sends $\mathsf{Proof}_0 = \mathsf{Proof}$ to $\mathcal{A}$, otherwise she runs the randomization algorithm $\mathsf{RandProof}$ to get $\mathsf{Proof}' = \mathsf{Proof}_1$ that she sends to $\mathcal{A}$. The latter also computes $\tilde{\mathsf{Proof}}$ from Batch and $\{\mathsf{Proof}_l\}_l$, and she sends $\mathsf{Proof}_b, \tilde{\mathsf{Proof}}$ to $\mathcal{B}$. $\mathcal{B}$ can distinguish a proof computed by Dele from a proof computed by Rede, therefore she wins the game with non-negligible advantage, and so does $\mathcal{A}$ by outputting $\mathcal{B}$'s answer.

Lemma 8. *Unlinkability is satisfied thanks to the witness indistinguishability property of the non-membership proof system.*

Proof. Let us see that if there is an adversary, let us say $\mathcal{B}$, that breaks the unlinkability property, then we can build an adversary, denoted $\mathcal{A}$, that breaks the witness indistinguishability property of the proof system. First, $\mathcal{A}$ is given Para from the challenger, and she simulates the accumulator challenger by computing $\mathsf{pk}_{\mathsf{acc}}$, that she sends to $\mathcal{B}$. The

latter chooses y_0, y_1 that she sends to $\mathcal{A}$. $\mathcal{A}$ then creates the witnesses $\left\{\mathsf{Wit}^l_{y_0}, \mathsf{Wit}^l_{y_1}\right\}_l$ that she sends to the challenger, along with $\{\mathsf{Sta}_l\}_l$. The challenger picks $b \in \{0,1\}$ and computes $\{\mathsf{Proof}_l\}$ from $\left\{\mathsf{Wit}^l_{y_b}\right\}_l$ then she runs Batch to get Proof, that is sent to $\mathcal{A}$. The latter then computes a proof Proof_0 for y_0 and she sends to $\mathcal{B}$ $(\mathsf{Proof}, \mathsf{Proof}_0)$. $\mathcal{B}$ can distinguish a proof computed for y_0 from a proof computed by for y_1, therefore she wins the game with non-negligible advantage, and so does $\mathcal{A}$ by outputting $\mathcal{B}$'s answer.

Note 17. In [1], they proved that their accumulator satisfies unlinkability as they used a composable ZK proof system. Actually, only witness indistinguishability is required.

How to obtain delegability? The witness indistinguishability and randomizable property of proof systems are not enough to obtain an accumulator with delegatable non-membership proof as *delegability* cannot be proven. To solve this issue, [2] uses a primitive they introduced: homomorphic proofs.

Definition 10. ***Homomorphic proofs [2].*** *Let* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verif})$ *be a proof system for a relation* R *and* $\mathsf{Para} \leftarrow \mathsf{Setup}(\lambda)$. *Consider a subset* Π *of all* $(\mathsf{Sta}, \mathsf{Wit}, \mathsf{Proof})$ *such that* $(\mathsf{Para}, \mathsf{Sta}, \mathsf{Wit}) \in R$ *and* $\mathsf{Verif}(\mathsf{Para}, \mathsf{Sta}, \mathsf{Proof}) = 1$*, and an operation* $+_\Pi : \Pi \times \Pi \to \Pi$. Π *is a set of homomorphic proofs if* $(\Pi, +_\Pi)$ *satisfies* ***closure****,* ***associativity*** *and* ***commutativity****. Consider an* $I_\Pi = (\mathsf{Sta}_0, \mathsf{Wit}_0, \mathsf{Proof}_0) \in \Pi$. Π *is a set of strongly homomorphic proofs if* $(\Pi, +_\Pi, I_\Pi)$ *forms an Abelian group where* I_Π *is the identity element.*

Lemma 9. *Delegability is satisfied thanks to the homomorphic proofs and the randomizable property of the non-membership proof system.*

Proof. Thanks to the homomorphic property, the proof output by $\mathsf{CompNMProof}$ is a valid proof for statement $\mathsf{Sta} = (\mathsf{acc}_\mathcal{X}, \mathsf{aux})$. Plus, as $\mathsf{CompNMProof}$ is using $\mathsf{RandProof}$ to randomize the computed proof, the proof is indistinguishable from a proof computed using the proof system Prove algorithm, for statement $\mathsf{Sta} = (\mathsf{acc}_\mathcal{X}, \mathsf{aux})$.

Conclusion. To obtain an accumulator scheme that has delegatable proofs, the used proof systems must: **i)** satisfy witness indistinguishability, **ii)** be randomizable, **iii)** have homomorphic proofs, and **iv)** support batching techniques. The last two points are the most complicated to obtain. Indeed, currently (as far as we know) there is only one proof system proven to have homomorphic proofs: Groth Sahai proofs. However, this holds only if some conditions on parameters Para, statements Sta and witnesses Wit are satisfied, such as the fact that witnesses and statements must have some constant parts. Quite the same goes for batching techniques: Groth-Sahai proofs support batching on some conditions only. Taking all that into account it seems that not all accumulators can be added delegation property and thus providing a generic construction is not possible.

References

1. Acar, T., Nguyen, L.: Revocation for delegatable anonymous credentials. Technical Repost MSR-TR-2010-170, Microsoft Research (2010)

2. Acar, T., Nguyen, L.: Revocation for delegatable anonymous credentials. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 423–440. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19379-8_26
3. Asano, T.: A revocation scheme with minimal storage at receivers. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 433–450. Springer, Heidelberg (Dec 2002). https://doi.org/10.1007/3-540-36178-2_27
4. Au, M.H., Tsang, P.P., Susilo, W., Mu, Y.: Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 295–308. Springer, Heidelberg (Apr 2009). https://doi.org/10.1007/978-3-642-00862-7_20
5. Au, M.H., Wu, Q., Susilo, W., Mu, Y.: Compact e-cash from bounded accumulator. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 178–195. Springer, Heidelberg (Feb 2007). https://doi.org/10.1007/11967668_12
6. Ayebie, E.B., Souidi, E.M.: New code-based cryptographic accumulator and fully dynamic group signature. DCC **90**(12), 2861–2891 (2022). https://doi.org/10.1007/s10623-022-01007-5
7. Baldimtsi, F., Camenisch, J., Dubovitskaya, M., Lysyanskaya, A., Reyzin, L., Samelin, K., Yakoubov, S.: Accumulators with applications to anonymity-preserving revocation. Cryptology ePrint Archive, Paper 2017/043 (2017), https://eprint.iacr.org/2017/043, https://eprint.iacr.org/2017/043
8. Baldimtsi, F., Camenisch, J., Dubovitskaya, M., Lysyanskaya, A., Reyzin, L., Samelin, K., Yakoubov, S.: Accumulators with applications to anonymity-preserving revocation. pp. 301–315 (04 2017). https://doi.org/10.1109/EuroSP.2017.13
9. Baldimtsi, F., Canetti, R., Yakoubov, S.: Universally composable accumulators. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 638–666. Springer, Heidelberg (Feb 2020). https://doi.org/10.1007/978-3-030-40186-3_27
10. Baldimtsi, F., Karantaidou, I., Raghuraman, S.: Oblivious accumulators. Cryptology ePrint Archive, Paper 2023/1001 (2023), https://eprint.iacr.org/2023/1001, https://eprint.iacr.org/2023/1001
11. Baldimtsi, F., Karantaidou, I., Raghuraman, S.: Oblivious accumulators. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography – PKC 2024. pp. 99–131. Springer Nature Switzerland, Cham (2024)
12. Bari, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_33
13. Barthoulot, A., Blazy, O., Canard, S.: Locally verifiable signatures and cryptographic accumulators: different names, same thing?
14. Barthoulot, A., Blazy, O., Canard, S.: Dually computable cryptographic accumulators and their application to attribute based encryption. Cryptology ePrint Archive, Paper 2023/1277 (2023), https://eprint.iacr.org/2023/1277, https://eprint.iacr.org/2023/1277
15. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: Helleseth, T. (ed.) EUROCRYPT'93. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (May 1994). https://doi.org/10.1007/3-540-48285-7_24
16. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to IOPs and stateless blockchains. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 561–586. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_20

17. Boneh, D., Corrigan-Gibbs, H.: Bivariate polynomials modulo composites and their applications. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 42–62. Springer, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45611-8_3
18. Buldas, A., Laud, P., Lipmaa, H.: Accountable certificate management using undeniable attestations. In: Gritzalis, D., Jajodia, S., Samarati, P. (eds.) ACM CCS 2000. pp. 9–17. ACM Press (Nov 2000). https://doi.org/10.1145/352600.352604
19. Buldas, A., Laud, P., Lipmaa, H.: Eliminating counterevidence with applications to accountable certificate management. Journal of Computer Security **10**, 273–296 (08 2002). https://doi.org/10.3233/JCS-2002-10304
20. Camacho, P., Hevia, A., Kiwi, M.A., Opazo, R.: Strong accumulators from collision-resistant hashing. In: Wu, T.C., Lei, C.L., Rijmen, V., Lee, D.T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 471–486. Springer, Heidelberg (Sep 2008)
21. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00468-1_27
22. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_5
23. Campanelli, M., Fiore, D., Han, S., Kim, J., Kolonelos, D., Oh, H.: Succinct zero-knowledge batch proofs for set accumulators. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 455–469. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560677
24. Canard, S., Gouget, A.: Multiple denominations in e-cash with compact transaction data. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 82–97. Springer, Heidelberg (Jan 2010)
25. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (Feb / Mar 2013). https://doi.org/10.1007/978-3-642-36362-7_5
26. Chen, J., Lim, H.W., Ling, S., Wang, H., Wee, H.: Shorter IBE and signatures via asymmetric pairings. In: Abdalla, M., Lange, T. (eds.) PAIRING 2012. LNCS, vol. 7708, pp. 122–140. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-36334-4_8
27. Couteau, G., Lipmaa, H., Parisella, R., Ødegaard, A.T.: Efficient nizks for algebraic sets. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021. pp. 128–158. Springer International Publishing, Cham (2021)
28. Damgard, I., Triandopoulos, N.: Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538 (2008), http://eprint.iacr.org/2008/538
29. Damgard, I., Triandopoulos, N.: Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Paper 2008/538 (2008), https://eprint.iacr.org/2008/538, https://eprint.iacr.org/2008/538
30. Derler, D., Hanser, C., Slamanig, D.: Revisiting cryptographic accumulators, additional properties and relations to other primitives. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 127–144. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-319-16715-2_7
31. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_36

32. Fazio, N., Nicolosi, A.: Cryptographic accumulators: Definitions, constructions and applications (2002)
33. Galbraith, S., Hess, F., Vercauteren, F.: Aspects of pairing inversion. IEEE Transactions on Information Theory **54**, 5719–5728 (01 2008)
34. Gentry, C., Ramzan, Z.: RSA accumulator based broadcast encryption. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 73–86. Springer, Heidelberg (Sep 2004)
35. Ghosh, E., Ohrimenko, O., Papadopoulos, D., Tamassia, R., Triandopoulos, N.: Zero-knowledge accumulators and set algebra. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 67–100. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_3
36. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_19
37. Jhanwar, M.P., Safavi-Naini, R.: Compact accumulator using lattices. In: Chakraborty, R.S., Schwabe, P., Solworth, J. (eds.) Security, Privacy, and Applied Cryptography Engineering. pp. 347–358. Springer International Publishing, Cham (2015)
38. Jhanwar, M.P., Tiwari, P.R.: Trading accumulation size for witness size: A merkle tree based universal accumulator via subset differences. Cryptology ePrint Archive, Paper 2019/1186 (2019), https://eprint.iacr.org/2019/1186, https://eprint.iacr.org/2019/1186
39. Jia, H., Chen, Y., Lan, J., Huang, K., Wang, J.: Efficient revocable hierarchical identity-based encryption using cryptographic accumulators. International Journal of Information Security (2018)
40. Lewko, A.B., Waters, B.: Unbounded HIBE and attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_30
41. Li, F., Hu, Y., Zhang, C.: An identity-based signcryption scheme for multi-domain ad hoc networks. In: Katz, J., Yung, M. (eds.) ACNS 07. LNCS, vol. 4521, pp. 373–384. Springer, Heidelberg (Jun 2007). https://doi.org/10.1007/978-3-540-72738-5_24
42. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) ACNS 07. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (Jun 2007). https://doi.org/10.1007/978-3-540-72738-5_17
43. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. International Conference on the Theory and Applications of Cryptographic Techniques (2016)
44. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 1–31. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_1
45. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) ICALP 2016. LIPIcs, vol. 55, pp. 30:1–30:14. Schloss Dagstuhl (Jul 2016). https://doi.org/10.4230/LIPIcs.ICALP.2016.30
46. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Lattice-based group signatures: Achieving full dynamicity with ease. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 17. LNCS, vol. 10355, pp. 293–312. Springer, Heidelberg (Jul 2017). https://doi.org/10.1007/978-3-319-61204-1_15

47. Lipmaa, H.: Secure accumulators from euclidean rings without trusted setup. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 12. LNCS, vol. 7341, pp. 224–240. Springer, Heidelberg (Jun 2012). https://doi.org/10.1007/978-3-642-31284-7_14
48. Lipmaa, H., Fauzi, P., Zhang, B.: Efficient non-interactive zero knowledge arguments for set operations. International Conference on Financial Cryptography and Data Security (2014)
49. Lipmaa, H., Parisella, R.: Set (non-)membership nizks from determinantal accumulators. Cryptology ePrint Archive, Paper 2022/1570 (2022), https://eprint.iacr.org/2022/1570, https://eprint.iacr.org/2022/1570
50. Mahabir, J., Reihaneh, S.N.: Compact accumulator using lattices. International Conference on Security, Privacy, and Applied Cryptography Engineering (2015)
51. de Meer, H., Liedel, M., Pohls, H.C., Posegga, J.: Indistinguishability of one-way accumulators. Technical Report MIP-1210, Faculty of Computer Science and Mathematics (FIM), University of Passau (2012)
52. de Meer, H., Pohls, H.C., Posegga, J., Samelin, K.: Redactable signature schemes for trees with signer-controlled non-leaf-redactions. E-Business and Telecommunications (2014)
53. Nguyen, K., Tang, H., Wang, H., Zeng, N.: New code-based privacy-preserving cryptographic constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 25–55. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34621-8_2
54. Nguyen, L.: Accumulators from bilinear pairings and applications. CT-RSA (2005)
55. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (Feb 2005). https://doi.org/10.1007/978-3-540-30574-3_19
56. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (Dec 2009). https://doi.org/10.1007/978-3-642-10366-7_13
57. Papamanthou, C., Shi, E., Tamassia, R., Yi, K.: Streaming authenticated data structures. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 353–370. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_22
58. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Optimal verification of operations on dynamic sets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 91–110. Springer, Heidelberg (Aug 2011). https://doi.org/10.1007/978-3-642-22792-9_6
59. Ren, Y., Liu, X., Wu, Q., Wang, L., Zhang, W.: Cryptographic accumulator and its application: A survey. Security and Communication Networks **2022**, 1–13 (03 2022). https://doi.org/10.1155/2022/5429195
60. Reyzin, L., Yakoubov, S.: Efficient asynchronous accumulators for distributed pki. Internarional Conference on Security and Cryptography fo Networks (2016)
61. Reyzin, L., Yakoubov, S.: Efficient asynchronous accumulators for distributed PKI. In: Zikas, V., De Prisco, R. (eds.) SCN 16. LNCS, vol. 9841, pp. 292–309. Springer, Heidelberg (Aug / Sep 2016). https://doi.org/10.1007/978-3-319-44618-9_16
62. Sander, T.: Efficient accumulators without trapdoor extended abstracts. In: Varadharajan, V., Mu, Y. (eds.) ICICS 99. LNCS, vol. 1726, pp. 252–262. Springer, Heidelberg (Nov 1999)
63. Tomescu, A., Bhupatiraju, V., Papadopoulos, D., Papamanthou, C., Triandopoulos, N., Devadas, S.: Transparency logs via append-only authenticated dictionaries. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 1299–1316. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3345652
64. Tsudik, G., Xu, S.: Accumulating composites and improved group signing. In: Laih, C.S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 269–286. Springer, Heidelberg (Nov / Dec 2003). https://doi.org/10.1007/978-3-540-40061-5_16

65. Wang, P., Wang, H., Pieprzyk, J.: A new dynamic accumulator for batch updates. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 07. LNCS, vol. 4861, pp. 98–112. Springer, Heidelberg (Dec 2008)
66. Wang, X., Chow, S.S.M.: Cross-domain access control encryption: Arbitrary-policy, constant-size, efficient. In: 2021 IEEE Symposium on Security and Privacy. pp. 748–761. IEEE Computer Society Press (May 2021). https://doi.org/10.1109/SP40001.2021.00023
67. Yu, Z., Au, M.H.A., Yang, R., Lai, J., Xu, Q.: Lattice-based universal accumulator with non-membership arguments. In: Australasian Conference on Information Security and Privacy (2018), https://api.semanticscholar.org/CorpusID:49642792
68. Zhao, Y., Yang, S., Huang, X.: Lattice-based dynamic universal accumulator: Design and application. Computer Standards and Interfaces **89**, 103807 (2023). https://doi.org/https://doi.org/10.1016/j.csi.2023.103807, https://www.sciencedirect.com/science/article/pii/S0920548923000880

A Assumptions

In this section we prove that assumption introduced in Section 4 can be reduced to CDH in $\mathbb{G}_2$, when considering type 2 pairings.

Definition 11. ***Computational Diffie-Hellman assumption in*** $\mathbb{G}_2$***.****(CDH) Let* $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ *be an asymmetric bilinear pairing group. The CDH assumption in* $\mathbb{G}_2$ *states that for random* $a, b \in \mathbb{Z}_p$*, given* (Γ, g_2^a, g_2^b) *it is hard to compute* g_2^{ab}*.*

To do the reduction, we need an intermediate assumption.

Definition 12. ***Intermediate assumption.*** *Let* $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ *be an asymmetric bilinear pairing group. The intermediate assumption states that for random* $a, b \in \mathbb{Z}_p$*, given* $(\Gamma, g_1^a, g_2^a, g_2^b)$ *it is hard to compute* g_1^{ab}*.*

Theorem 6. *If CDH in* $\mathbb{G}_2$ *holds, then the intermediate assumption holds.*

Proof. We prove the contrapositive. Let $\mathcal{B}$ be an adversary that breaks the intermediate assumption with non-negligible advantage. We build $\mathcal{A}$ that uses $\mathcal{B}$ to break CDH in $\mathbb{G}_2$. $\mathcal{A}$ is given (Γ, g_2^a, g_2^b). As we are using type 2 pairings, there exists ϕ from $\mathbb{G}_2$ to $\mathbb{G}_1$. Then $\mathcal{A}$ gives to $\mathcal{B}$: $(\Gamma, \phi(g_2^a) = g_1^a, g_2^a, g_2^b)$. The latter answers with g_1^{ab} and $\mathcal{A}$ returns $\phi(g_1^{ab}) = g_2^{ab}$ as her answers. Notice that $\mathcal{A}$'s advantage is equal to $\mathcal{B}$'s advantage, therefore it is non-negligible.

Note 18. The intermediate assumption has a unique solution. Indeed we can rewrite the assumption as a fixed pairing inversion problem [33] as its aim is to find, given $g_2 \in \mathbb{G}_2$ and $e(g_1, g_2)^{ab} \in \mathbb{G}_T$ (computed from the other inputs of the assumption), $t \in \mathbb{G}_1$ such that $e(t, g_2) = e(g_1, g_2)^{ab}$. Then as stated in [33], as e is non-degenerate and the groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic of prime order p, then solution to the above problem is unique.

Before to prove reduction of the fixed argument dual pairing vector spaces inversion assumption to CDH in $\mathbb{G}_2$, we prove the following lemma.

Lemma 10. *The fixed argument dual pairing vector spaces inversion assumption has a unique solution.*

Proof. Breaking the assumption means to find an element $\boldsymbol{t} = (g_1^{t_1}, g_1^{t_2}) \in \mathbb{G}_1^2$ such that

$$\begin{cases} e(\boldsymbol{t}, g_2^{\boldsymbol{d}_1^*}) = e(g_1, g_2)^{\psi} \\ e(\boldsymbol{t}, g_2^{\boldsymbol{d}_2^*}) = 1 \end{cases} \tag{1}$$

where $\psi \in \mathbb{Z}_p$ and 1 is the identity element of $\mathbb{G}_T$. The above system can be rewritten as

$$\begin{cases} e(g_1^{t_1}, g_2^{d_{1,1}^*}) \cdot e(g_1^{t_2}, g_2^{d_{1,2}^*}) = e(g_1, g_2)^{\psi} \\ e(g_1^{t_1}, g_2^{d_{2,1}^*}) \cdot e(g_1^{t_2}, g_2^{d_{2,2}^*}) = 1 \end{cases} \tag{2}$$

In the exponent, the system becomes

$$\begin{cases} t_1 d_{1,1}^* + t_2 d_{1,2}^* = \psi \\ t_1 d_{2,1}^* + t_2 d_{2,2}^* = 0 \end{cases} \tag{3}$$

and has a unique solution if $d_{1,1}^* d_{2,2}^* - d_{2,1}^* d_{1,2}^* \neq 0$. By case-based reasoning we have that $d_{1,1}^* d_{2,2}^* - d_{2,1}^* d_{1,2}^* = 0$ if

- $(d_{1,1}^*, d_{1,2}^*)$ or $(d_{2,1}^*, d_{2,2}^*)$ is equal to $(0, 0)$. This is not possible by dual pairing vector spaces definition.
- $(d_{1,1}^*, d_{1,2}^*)$ and $(d_{2,1}^*, d_{2,2}^*)$ are respectively equals to either $(d_{1,1}^*, 0)$ and $(d_{2,1}^*, 0)$ or $(0, d_{1,2}^*)$ and $(0, d_{2,2}^*)$. In these cases the system (3) becomes $\begin{cases} t_1 d_{1,1}^* = \psi \\ t_1 d_{2,1}^* = 0 \end{cases}$ or $\begin{cases} t_2 d_{1,2}^* = \psi \\ t_2 d_{2,2}^* = 0 \end{cases}$ which does not have a solution. By definition of dual pairing vector spaces, this is not possible.
- $(d_{1,1}^*, d_{1,2}^*) = (d_{2,1}^*, d_{2,2}^*)$ which is not possible by definition of dual pairing vector spaces.
- $(d_{2,1}^*, d_{2,2}^*) = (d_{1,2}^*, d_{1,1}^*)$. In this case the equation $d_{1,1}^* d_{2,2}^* - d_{2,1}^* d_{1,2}^* = 0$ that can be rewritten as $d_{1,1}^* d_{2,2}^* = d_{2,1}^* d_{1,2}^*$ becomes $d_{1,1}^{*2} = d_{1,2}^{*2}$. We now have two cases:
  - either $d_{1,1}^* = d_{1,2}^*$ thus in this case $(d_{1,1}^*, d_{1,2}^*) = (d_{2,1}^*, d_{2,2}^*)$ which is not possible by definition of dual pairing vector spaces;
  - or $d_{1,1}^* = -d_{1,2}^*$ thus in this case the system 3 becomes $\begin{cases} -d_{1,2}^* t_1 + d_{1,2}^* t_2 = 0 \\ d_{1,2}^* t_1 - d_{1,2}^* t_2 = 0 \end{cases}$ and thus $\begin{cases} 0 = \psi \\ d_{1,2}^* t_1 = d_{1,2}^* t_2 \end{cases}$ which is not possible as $\psi \neq 0$.

Thus, by construction $d_{1,1}^* d_{2,2}^* - d_{2,1}^* d_{1,2}^* \neq 0$ and the above system as a unique solution: $\boldsymbol{t} = g_1^{\boldsymbol{d}_1}$.

Now we can reduce the fixed argument dual pairing vector spaces inversion (FA-DPVS-I) assumption to the intermediate assumption.

Theorem 7. *If the intermediate assumption holds, then fixed argument dual pairing vector spaces inversion assumption holds.*

Proof. We prove the contrapositive. Let $\mathcal{B}$ be an adversary that breaks the FA-DPVS-I assumption with non-negligible advantage. We build $\mathcal{A}$ that uses $\mathcal{B}$ to break the intermediate assumption. $\mathcal{A}$ is given $(\Gamma, g_1^a, g_2^a, g_2^b)$. $\mathcal{A}$ runs $\mathsf{Dual}(\mathbb{Z}_p^2)$ to get $(\mathbb{B}, \mathbb{B}^*)$ dual orthonormal bases of dimension 2. Then $\mathcal{A}$ defines new orthonormal bases $(\mathbb{D}, \mathbb{D}^*)$ as follows: $\boldsymbol{d}_1 = ab\boldsymbol{b}_1$, $\boldsymbol{d}_2 = a\boldsymbol{b}_2$, $\boldsymbol{d}_1^* = \boldsymbol{b}_1^*$ and $\boldsymbol{d}_2^* = b\boldsymbol{b}_2^*$. We easily notice that $(\mathbb{D}, \mathbb{D}^*)$ are dual orthonormal. Then $\mathcal{A}$ gives $(\Gamma, g_1^{\boldsymbol{d}_2}, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_2^*})$ to $\mathcal{B}$. Notice that $g_1^{\boldsymbol{d}_2} = (g_1^a)^{\boldsymbol{b}_2}$, $g_2^{\boldsymbol{d}_1^*} = g_2^{\boldsymbol{b}_1^*}$ and $g_2^{\boldsymbol{d}_2^*} = (g_2^b)^{\boldsymbol{d}_2^*}$. $\mathcal{B}$ answers with $\boldsymbol{t} = (t_1, t_2) \in \mathbb{G}_1^2$, which is equal to $g_1^{\boldsymbol{d}_1} \in \mathbb{G}_1^2$ by uniqueness of the solution in FAP-DPVS-I assumption, and can be rewrite as $(g_1^{\boldsymbol{d}_{1,1}}, g_1^{\boldsymbol{d}_{1,2}})$. By construction, it is equal to $(g_1^{ab\boldsymbol{b}_{1,1}}, g_1^{ab\boldsymbol{b}_{1,2}})$ and $\mathcal{A}$ can returns $t_1^{(\boldsymbol{b}_{1,1})^{-1}}$ as her answers as it is equal to g_1^{ab}.

Finally, we can prove the following theorem.

Theorem 8. *If the computational Diffie-Hellman assumption in $\mathbb{G}_2$ holds, then the fixed argument dual pairing vector spaces inversion assumption holds.*

The proof of the theorem is done by combining Theorem 6 and Theorem 7.