Revisiting Cryptographic Accumulators, Additional Properties and Relations to Other Primitives

David Derler(✉), Christian Hanser, and Daniel Slamanig

Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology (TUG), Inffeldgasse 16a, 8010 Graz, Austria
{david.derler,christian.hanser,daniel.slamanig}@tugraz.at

Abstract. Cryptographic accumulators allow to accumulate a finite set of values into a single succinct accumulator. For every accumulated value, one can efficiently compute a witness, which certifies its membership in the accumulator. However, it is computationally infeasible to find a witness for any non-accumulated value. Since their introduction, various accumulator schemes for numerous practical applications and with different features have been proposed. Unfortunately, to date there is no unifying model capturing all existing features. Such a model can turn out to be valuable as it allows to use accumulators in a black-box fashion.

To this end, we propose a unified formal model for (randomized) cryptographic accumulators which covers static and dynamic accumulators, their universal features and includes the notions of undeniability and indistinguishability. Additionally, we provide an exhaustive classification of all existing schemes. In doing so, it turns out that most accumulators are distinguishable. Fortunately, a simple, light-weight generic transformation allows to make many existing dynamic accumulator schemes indistinguishable. As this transformation, however, comes at the cost of reduced collision freeness, we additionally propose the first indistinguishable scheme that does not suffer from this shortcoming. Finally, we employ our unified model for presenting a black-box construction of commitments from indistinguishable accumulators as well as a black-box construction of indistinguishable, undeniable universal accumulators from zero-knowledge sets. Latter yields the first universal accumulator construction that provides indistinguishability.

1 Introduction

A (static) cryptographic accumulator scheme allows to accumulate a finite set $\mathcal{X} = \{x_1, \ldots, x_n\}$ into a succinct value $\mathsf{acc}_{\mathcal{X}}$, the so called accumulator. For every element $x_i \in \mathcal{X}$, one can efficiently compute a so called witness wit_{x_i} to certify

The authors have been supported by the European Commission through project FP7-FutureID, grant agreement number 318424. An extended version of this paper is available in the IACR Cryptology ePrint Archive.

K. Nyberg (ed.): CT-RSA 2015, LNCS 9048, pp. 127–144, 2015.
DOI: 10.1007/978-3-319-16715-2_7

the membership of x_i in $\mathsf{acc}_{\mathcal{X}}$. However, it should be computationally infeasible to find a witness for any non-accumulated value $y \notin \mathcal{X}$ (*collision freeness*). Dynamic accumulators are an extension that allows to dynamically add/delete values to/from a given accumulator and to update existing witnesses accordingly (without the need to fully recompute these values on each change of the accumulated set). Besides providing membership witnesses, universal accumulators also support non-membership witnesses for values $y \notin \mathcal{X}$. Here, collision freeness also covers that it is computationally infeasible to create non-membership witnesses for values $x_i \in \mathcal{X}$. Over time, further security properties, that is, *undeniability* and *indistinguishability* have been proposed. Undeniability is specific to universal accumulators and says that it should be computationally infeasible to compute two contradicting witnesses for $z \in \mathcal{X}$ and $z \notin \mathcal{X}$. Indistinguishability says that neither the accumulator nor the witnesses leak information about the accumulated set $\mathcal{X}$ and, thus, requires randomized accumulator schemes.

Applications: Accumulators were originally proposed for timestamping purposes [5], i.e., to record the existence of a value at a particular point in time. Over time, other applications such as membership testing, distributed signatures, accountable certificate management [7] and authenticated dictionaries [22] have been proposed. Accumulators are also used as building block in redactable [33,34], sanitizable [13], P-homomorphic signatures [2], anonymous credentials [38], group signatures [39], privacy-preserving data outsourcing [37] as well as for authenticated data structures [21]. Moreover, accumulator schemes that allow to prove the knowledge of a (non-membership) witness for an unrevealed value in zero-knowledge (introduced for off-line e-cash in [36]) are now widely used for revocation of group signatures and anonymous credentials [12]. Quite recently, accumulators were also used in Zerocoin [28], an anonymity extension to the Bitcoin cryptocurrency.

Since their introduction, numerous accumulator schemes with somewhat different features have been proposed. Basically, the major lines of work are schemes in hidden order groups (RSA), known order groups (DL) and hash-based constructions (which may use, but typically do not require number theoretic assumptions).

Hidden Order Groups: The original RSA-based scheme of Benaloh and de Mare [5] has been refined by Baric and Pfitzmann [4], who strengthen the original security notion to *collision freeness*. In [35], Sander proposed to use RSA moduli with unknown factorization to construct trapdoor-free accumulators. Camenisch and Lysyanskaya [12] extended the scheme in [4] with capabilities to dynamically add/delete values to/from the accumulator, which constituted the first *dynamic accumulator* scheme. Their scheme also supports *public updates* of existing witnesses, that is, updates without the knowledge of any trapdoor. Later, Li et al. [24] added support for non-membership witnesses to [12] and, therefore, obtained *universal dynamic accumulators*. They also proposed an optimization for more efficient updates of non-membership witnesses, for which, however, weaknesses have been identified later [26,32]. Lipmaa [25] generalized RSA accumulators to modules over Euclidean rings. In all aforementioned schemes, the accumulation domain is restricted to primes in order to guarantee collision freeness. In [39], Tsudik and Xu proposed a variation of [12], which allows to accumulate semiprimes. This yields a collision-free accumulator under the assumption that the used semiprimes are hard to factor and their factorization is not publicly known. Moreover, in [40] an accumulator scheme that allows to accumulate arbitrary integers and supports batch updates of witnesses has been proposed. Yet, this scheme was broken in [9].

Known Order Groups: In [29], Nguyen proposed a dynamic accumulator scheme which works in pairing-friendly groups of prime order p. It is secure under the t-SDH assumption and allows to accumulate up to t values from the domain $\mathbb{Z}_p$. Later, Damgård and Triandopoulos [16] as well as Au et al. [3] extended Nguyen's scheme with universal features. Quite recently, Acar and Nguyen [1] eliminated the upper bound t on the number of accumulated elements of the t-SDH accumulator. To this end, they use a set of accumulators, each containing a subset of the whole set to be accumulated. An alternative accumulator scheme for pairing friendly groups of prime order has been introduced by Camenisch et al. [11]. It supports public updates of witnesses and the accumulator and its security relies on the t-DHE assumption.

Hash-Based Constructions: Buldas et al. [7,8] presented the very first universal dynamic accumulator that satisfies *undeniability* (termed as undeniable attester and formalized in context of accumulators in [25]). Their construction is based on collision-resistant hashing and the use of hash-trees. Another hash-tree based construction of a universal accumulator that satisfies a notion similar to undeniability has been proposed in [10] (the scheme is called a strong universal accumulator). Quite recently, another accumulator based on hash-trees, which uses commitments based on bivariate polynomials modulo RSA composites as a collision-resistant hash function, has been introduced in [6]. For the sake of completeness, we also mention the construction of static accumulators in the random oracle model based on Bloom filters, proposed by Nyberg [30,31].

Contribution: The contributions of this paper are as follows:

– While some papers [3–5,12,29] do not explicitly formalize accumulator schemes, formal definitions are given in [1,10,11,14,20,24,25,40]. However, these models are typically tailored to the functionalities of the respective scheme. While they widely match for the basic notion of (static) accumulators (with the exception of considering randomized accumulators), they differ when it comes to dynamic and universal accumulators. To overcome this issue, we propose a unified formal model for accumulators, which is especially valuable when treating accumulators in a black-box fashion. We, thereby, also include the notion of undeniability [7,8,25] and a strengthened version of the recent indistinguishability notion [17]. Besides, we also confirm the intuition and show that undeniability is a strictly stronger notion than collision freeness.

- We provide an exhaustive classification of existing accumulator schemes and show that most existing accumulator schemes are distinguishable in our model. To resolve this issue, we propose a simple, light-weight generic transformation that allows to add indistinguishability to existing dynamic accumulators and prove the security of the so-obtained schemes. As this transformation, however, comes at the cost of reduced collision freeness, we additionally propose the first indistinguishable scheme that does not suffer from this shortcoming. Note that due to the lack of space, the indistinguishable accumulator scheme is provided in the extended version of this paper.
- Since accumulators are somehow related to commitments to sets [19,23], commitments to vectors [14] and to zero-knowledge sets [27], it is interesting to study their relationship. Interestingly, we can formally show that indistinguishable accumulators imply non-interactive commitment schemes. Furthermore, we formally show that zero-knowledge sets imply indistinguishable, undeniable universal accumulators, yielding the first construction of such accumulators.

2 Preliminaries

By acc we denote an accumulator and if we want to make the accumulated set $\mathcal{X} = \{x_1, \ldots, x_n\}$ explicit, we write $\mathsf{acc}_{\mathcal{X}}$. Given an accumulator $\mathsf{acc}_{\mathcal{X}}$, a membership witness for an element $x_i \in \mathcal{X}$ is denoted by wit_{x_i}, whereas a non-membership witness for an element $y_j \notin \mathcal{X}$ is denoted by $\underline{\mathsf{wit}}_{y_j}$. The accumulator secret key (trapdoor) is denoted by $\mathsf{sk}_{\mathsf{acc}}$, while the public key is denoted by $\mathsf{pk}_{\mathsf{acc}}$. By $a \overset{R}{\leftarrow} A$, we denote that a is chosen uniformly at random from the set A.

A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ is called *negligible* if for all $c > 0$ there is a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. In the remainder of this paper, we use ϵ to denote such a negligible function.

3 A Unified Model for Cryptographic Accumulators

In the original sense, accumulator schemes were defined by the following properties (see, e.g., [12,24]). Thereby, $\mathcal{Z}_I$ represents the domain of values to be accumulated and $\mathcal{Z}_A$ the accumulator domain.

Efficient generation: There is an efficient probabilistic algorithm that, on input of a security parameter κ, defines a functionality $f : \mathcal{Z}_A \times \mathcal{Z}_I \rightarrow \mathcal{Z}_A$, i.e., generates the accumulator specific key pair ($\mathsf{sk}_{\mathsf{acc}}$, $\mathsf{pk}_{\mathsf{acc}}$) (where $\mathsf{sk}_{\mathsf{acc}}$ is a trapdoor for f).

Efficient evaluation: There is an efficient algorithm that computes $f(\mathsf{acc}, x)$.

Quasi-commutativity: It holds that $f(f(\mathsf{acc}, x_1), x_2) = f(f(\mathsf{acc}, x_2), x_1)$ $\forall x_1, x_2 \in \mathcal{Z}_I, \mathsf{acc} \in \mathcal{Z}_{\mathsf{A}}$.

Assuming that it is computationally infeasible to invert f without knowing $\mathsf{sk}_{\mathsf{acc}}$, the quasi-commutativity directly yields a way to define witnesses. For instance, $f(\mathsf{acc}, x_1)$ can serve as witness for the accumulation of x_2. Nonetheless, it is more meaningful to provide a more abstract algorithmic definition of accumulators as done subsequently, since there are several constructions that do not fit into this characterization (for instance, hash-tree constructions do not require the quasi-commutativity property).

Trusted vs. Non-Trusted Setup: Known accumulators that rely on number theoretic assumptions require a trusted setup, i.e., a TTP runs the setup algorithm Gen and discards the trapdoor $\mathsf{sk}_{\mathsf{acc}}$ afterwards. Here, access to $\mathsf{sk}_{\mathsf{acc}}$ allows to break collision freeness (and its stronger form: undeniability). Consequently, correctness of the accumulator scheme also needs to hold if $\mathsf{sk}_{\mathsf{acc}}$ is omitted in all algorithms, which is the case for all existing schemes. In contrast, in constructions relying on collision-resistant hash functions (not based on number theoretic assumptions) there is no trapdoor at all and, therefore, no trusted setup is required. In order to study number theoretic accumulators without trusted setup, Lipmaa [25] proposed a modified model which divides the Gen algorithm into a Setup and a Gen algorithm. In this model, the adversary can control the randomness used inside Setup and, thus, knows the trapdoor. Nevertheless, it can neither access nor influence the randomness of the Gen algorithm. This model, however, still requires a partially trusted setup and also does not fit to the known order group setting, which makes it not generally applicable.[1] Consequently, when considering the state of the art it seems most reasonable to define a security model with respect to a trusted setup as we will do subsequently. We emphasize that this model is compatible with all existing constructions. Nevertheless, it remains a challenging open issue to design accumulators based on standard assumptions which are secure without any trusted setup.

3.1 Definitions

In the following, we provide a definition for (static) accumulators, which we adapt from [20,40]. In contrast to previous models, we explicitly consider randomized accumulator schemes. Then, we extend this model in order to formalize dynamic accumulators. It is similar to [11,14], but avoids shortcomings such as missing private updates. Based on this, we define universal and universal dynamic accumulators and propose a suitable security model. Furthermore, we discuss undeniable and indistinguishable accumulators, give formalizations for these properties, and, investigate relationships between security properties.

[1] This model is tailored to the hidden order group setting, where Setup produces a composite modulus N. Gen chooses a random generator g of a large subgroup of $\mathbb{Z}_N^*$. Then, the adversary knows the factorization of N but does not control the choice of g. RSA accumulators are obviously insecure in this setting, but Lipmaa provides secure solutions based on modules over an Euclidean ring, which, however, rely on rather unstudied assumptions.

We call accumulators that have an upper bound t on the number of accumulated values *t-bounded accumulators* and *unbounded* otherwise. In order to model this, our Gen algorithm takes an additional parameter t, where $t = \infty$ is used to indicate that the accumulator is unbounded. For the sake of completeness, we model the algorithms such that they support an optional input of the trapdoor (denoted as $\mathsf{sk}^{\sim}_{\mathsf{acc}}$) since this often allows to make the algorithms more efficient. However, we stress that we consider the trusted setup model and, hence, adversaries are not given access to the trapdoor $\mathsf{sk}_{\mathsf{acc}}$. Consequently, if $\mathsf{sk}^{\sim}_{\mathsf{acc}}$ is set, the party running the algorithm needs to be fully trusted.

Definition 1 (Static Accumulator). *A static accumulator is a tuple of efficient algorithms* (Gen, Eval, $\mathsf{WitCreate}$, Verify) *which are defined as follows:*

$\mathsf{Gen}(1^\kappa, t)$: *This algorithm takes a security parameter κ and a parameter t. If $t \neq \infty$, then t is an upper bound on the number of elements to be accumulated. It returns a key pair* $(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$*, where* $\mathsf{sk}_{\mathsf{acc}} = \emptyset$ *if no trapdoor exists.*

$\mathsf{Eval}((\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}), \mathcal{X})$: *This (probabilistic)*[2] *algorithm takes a key pair* $(\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$ *and a set $\mathcal{X}$ to be accumulated and returns an accumulator* $\mathsf{acc}_{\mathcal{X}}$ *together with some auxiliary information* aux.

$\mathsf{WitCreate}((\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$: *This algorithm takes a key pair* $(\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$*, an accumulator* $\mathsf{acc}_{\mathcal{X}}$*, auxiliary information* aux *and a value x_i. It returns $\perp$, if $x_i \notin \mathcal{X}$, and a witness* wit_{x_i} *for x_i otherwise.*

$\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_{x_i}, x_i)$: *This algorithm takes a public key* $\mathsf{pk}_{\mathsf{acc}}$*, an accumulator* $\mathsf{acc}_{\mathcal{X}}$*, a witness* wit_{x_i} *and a value x_i. It returns* `true` *if* wit_{x_i} *is a witness for $x_i \in \mathcal{X}$ and* `false` *otherwise.*

Henceforth, we call an accumulator *randomized* if the Eval algorithm is probabilistic. Based on Definition 1, we can now formalize *dynamic accumulators*. We widely align our definitions with [20,40], but, in addition, we need to consider that the various dynamic accumulator schemes proposed so far differ regarding the *public updatability* of witnesses and the accumulator.

Definition 2 (Dynamic Accumulator). *A dynamic accumulator is a static accumulator that additionally provides efficient algorithms* ($\mathsf{Add}, \mathsf{Delete}, \mathsf{WitUpdate}$) *which are defined as follows:*

$\mathsf{Add}((\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$: *This algorithm takes a key pair* $(\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$*, an accumulator* $\mathsf{acc}_{\mathcal{X}}$*, auxiliary information* aux*, as well as a value x_i to be added. If $x_i \in \mathcal{X}$, it returns $\perp$. Otherwise, it returns the updated accumulator* $\mathsf{acc}_{\mathcal{X}'}$ *with $\mathcal{X}' \leftarrow \mathcal{X} \cup \{x_i\}$ and updated auxiliary information* aux'.

$\mathsf{Delete}((\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$: *This algorithm takes a key pair* $(\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$*, an accumulator* $\mathsf{acc}_{\mathcal{X}}$*, auxiliary information* aux*, as well as a value x_i to be removed. If $x_i \notin \mathcal{X}$, it returns $\perp$. Otherwise, it returns the updated accumulator* $\mathsf{acc}_{\mathcal{X}'}$ *with $\mathcal{X}' \leftarrow \mathcal{X} \setminus \{x_i\}$ and auxiliary information* aux'.

[2] If Eval is probabilistic, the internally used randomness is denoted as r. If we want to make the randomness used by the Eval algorithm explicit, we will write Eval_r.

$\mathsf{WitUpdate}((\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{wit}_{x_i}, \mathsf{aux}, x_j)$**:** *This algorithm takes a key pair* $(\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}})$*, a witness* wit_{x_i} *to be updated, auxiliary information* aux *and a value* x_j *which was added/deleted to/from the accumulator, where* aux *indicates addition or deletion. It returns an updated witness* wit'_{x_i} *on success and* $\perp$ *otherwise.*

Below, we define *universal accumulators* and emphasize that features provided by universal accumulators can be seen as supplementary features to both *static* and *dynamic* accumulators.

Definition 3 (Universal Accumulator). *A universal accumulator is a static or a dynamic accumulator with the following properties. For static accumulator schemes the algorithms* $\mathsf{WitCreate}$ *and* Verify *take an additional boolean parameter* type*, indicating whether the given witness is a membership* $(\mathsf{type} = 0)$ *or non-membership* $(\mathsf{type} = 1)$ *witness. For dynamic accumulator schemes this additionally applies to* $\mathsf{WitUpdate}$.

3.2 Security Model

Now, we introduce a security model for accumulators, which we adapt from [24] and further extend by undeniability and indistinguishability.

Classic Notion: A secure accumulator scheme is required to be *correct* and *collision-free*. Correctness says that for all honestly generated keys, all honestly computed accumulators and witnesses, the Verify algorithm will always return `true`. We stress that correctness also needs to hold when all algorithms are executed without $\mathsf{sk}_{\mathsf{acc}}$. Since the correctness property is straightforward, we omit its formal definition. Collision freeness informally states that it is neither feasible to find a witness for a non-accumulated value nor feasible to find a non-membership witness for an accumulated value. More formally:

Definition 4 (Collision Freeness). *A cryptographic accumulator of type* $\mathsf{t} \in \{\mathsf{static}, \mathsf{dynamic}\}$ *and* $\mathsf{u} \in \{\mathsf{universal}, \mathsf{non\text{-}universal}\}$ *is* collision-free*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{c} (\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Gen}(1^{\kappa}, t),\ \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{t}}, \mathcal{O}^{\mathsf{u}}\}, \\ (\mathsf{wit}^*_{x_i}/\underline{\mathsf{wit}}^*_{x_i}, x^*_i, \mathcal{X}^*, r^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}_{\mathsf{acc}}) : \\ (\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}^*, \mathsf{wit}^*_{x_i}, x^*_i, 0) = \mathtt{true}\ \wedge\ x^*_i \notin \mathcal{X}^*) \quad \vee \\ (\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}^*, \underline{\mathsf{wit}}^*_{x_i}, x^*_i, 1) = \mathtt{true}\ \wedge\ x^*_i \in \mathcal{X}^*) \end{array}\right] \leq \epsilon(\kappa),$$

where $\mathsf{acc}^* \leftarrow \mathsf{Eval}_{r^*}((\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}), \mathcal{X}^*)$ *and* $\mathcal{A}$ *has oracle access to* $\mathcal{O}^{\mathsf{t}}$ *and* $\mathcal{O}^{\mathsf{u}}$ *which are defined as follows:*

$$\mathcal{O}^{\mathsf{t}} := \begin{cases} \{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)}\} & \textit{if } \mathsf{t} = \mathsf{static}, \\ \{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)}, \mathcal{O}^{\mathsf{A}(\cdot,\cdot,\cdot,\cdot)}, \mathcal{O}^{\mathsf{D}(\cdot,\cdot,\cdot,\cdot)}\} & \textit{otherwise.} \end{cases}$$

$$\mathcal{O}^{\mathsf{u}} := \begin{cases} \{\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\cdot,\cdot)}, \mathcal{O}^{\underline{\mathsf{W}}(\cdot,\cdot,\cdot,\cdot)}\} & \textit{if } \mathsf{u} = \mathsf{universal}, \\ \{\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\cdot,\cdot)}\} & \textit{otherwise.} \end{cases}$$

Thereby, $\mathcal{O}^{\mathsf{E}}, \mathcal{O}^{\mathsf{A}}$ and $\mathcal{O}^{\mathsf{D}}$ represent the oracles for the algorithms $\mathsf{Eval}, \mathsf{Add}$, and Delete, respectively. An adversary is allowed to query them an arbitrary number of times. In case of randomized accumulators the adversary outputs randomness r^*, whereas r^* is omitted for deterministic accumulators. Likewise, the adversary can control the randomness r used by $\mathcal{O}^{\mathsf{E}}$ for randomized accumulators. Therefore, $\mathcal{O}^{\mathsf{E}}$ takes an additional parameter for r (which is missing for deterministic accumulators). The oracles $\mathcal{O}^{\mathsf{W}}$ and $\mathcal{O}^{\underline{\mathsf{W}}}$ allow the adversary to obtain membership witnesses for members and non-membership witnesses for non-members, respectively. Thereby, the environment keeps track of all oracle queries (and answers) and lets the respective oracle return $\perp$ if calls to it are not consistent with respect to previous queries. Furthermore, we assume that the adversary outputs either a membership witness $\mathsf{wit}^*_{x_i}$ or a non-membership witness $\underline{\mathsf{wit}}^*_{x_i}$ (denoted by $\mathsf{wit}^*_{x_i}/\underline{\mathsf{wit}}^*_{x_i}$). If the accumulator is non-universal, one simply omits the non-membership related parts.

One distinction to previous models is that we model (non-)membership witness generation via oracles. This way, we can ensure that security proofs take the simulation of (non-)membership witnesses into account, which is vital and could be overseen otherwise.

Definition 5 (Secure Accumulator). *A cryptographic accumulator is* secure *if it is correct and collision-free.*

Undeniable accumulators: In [25], Lipmaa formalized undeniability for accumulators. A universal accumulator is *undeniable* if it is computationally infeasible to find a membership as well as a non-membership witness for the same value – independently of whether it is contained in an accumulator or not. More formally undeniability is defined as:

Definition 6 (Undeniability). *A universal cryptographic accumulator of type* $\mathsf{t} \in \{\mathsf{static}, \mathsf{dynamic}\}$ *is* undeniable*, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[\begin{array}{c}(\mathsf{sk_{acc}}, \mathsf{pk_{acc}}) \leftarrow \mathsf{Gen}(1^\kappa, t), (\mathsf{wit}^*_{x_i}, \underline{\mathsf{wit}}^*_{x_i}, x_i^*, \mathsf{acc}^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{t}}}(\mathsf{pk_{acc}}) : \\ \mathsf{Verify}(\mathsf{pk_{acc}}, \mathsf{acc}^*, \mathsf{wit}^*_{x_i}, x_i^*, 0) = \mathtt{true} \ \wedge \\ \mathsf{Verify}(\mathsf{pk_{acc}}, \mathsf{acc}^*, \underline{\mathsf{wit}}^*_{x_i}, x_i^*, 1) = \mathtt{true}\end{array}\right] \leq \epsilon(\kappa),$$

where, $\mathcal{A}$ has oracle access to $\mathcal{O}^{\mathsf{t}}$ which is defined as follows:

$$\mathcal{O}^{\mathsf{t}} := \begin{cases} \{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)}, \mathcal{O}^{\mathsf{W}(\cdot,\cdot,\cdot,\cdot)}, \mathcal{O}^{\underline{\mathsf{W}}(\cdot,\cdot,\cdot,\cdot)}\} & \textit{if } \mathsf{t} = \mathsf{static}, \\ \{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)}, \mathcal{O}^{\mathsf{A}(\cdot,\cdot,\cdot,\cdot)}, \mathcal{O}^{\mathsf{D}(\cdot,\cdot,\cdot,\cdot)}, \mathcal{O}^{\mathsf{W}(\cdot,\cdot,\cdot,\cdot)}, \mathcal{O}^{\underline{\mathsf{W}}(\cdot,\cdot,\cdot,\cdot)}\} & \textit{otherwise.} \end{cases}$$

Notice that the definition of the oracles is as in the definition of collision freeness for universal accumulators.

Definition 7. *A universal accumulator is* undeniable *if it is a secure accumulator satisfying the undeniability property.*

Indistinguishable Accumulators: Li et al. [24] pointed out informally (without giving any formalizations) that the accumulation of an additional random value from the accumulation domain renders guessing the accumulated set infeasible. Later, de Meer et al. [17] tried to formalize this intuition via an additional *indistinguishability* property. Unfortunately, there are some issues with their notion. Firstly, it only covers static accumulators and, secondly, indistinguishability in the vein of [24] weakens collision resistance. Basically, one can easily generate a membership witness for the random value. Secondly, the security game in [17] allows to prove indistinguishability of deterministic accumulators, which are clearly not indistinguishable. In particular, the random value is chosen and accumulated within the security game. However, this non-determinism is not required to be part of the accumulator construction itself. Consequently, a deterministic accumulator can satisfy this notion while being trivially distinguishable. From this, we conclude that the non-determinism must be intrinsic to the Eval algorithm.[3]

There are several ways to turn a deterministic scheme into a randomized one. As already discussed, indistinguishability can be achieved by adding a random value from the accumulation domain. Aside from this, it can also be obtained by randomizing the Eval algorithm without modifying the set $\mathcal{X}$ (as, for instance, done in the extended version of this paper). Apparently, the latter option depends on the specific accumulator scheme, whereas the shortcomings in [17] can be addressed by introducing a generic transformation for the former approach (cf. Transformation 1).

Definition 8 (Indistinguishability). *A cryptographic accumulator of type* $\mathsf{t} \in \{\mathsf{static}, \mathsf{dynamic}\}$ *and* $\mathsf{u} \in \{\mathsf{universal}, \mathsf{non\text{-}universal}\}$ *is* indistinguishable*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that:*

$$\mathsf{Pr}\left[\begin{array}{c}(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Gen}(1^{\kappa}, t),\ b \xleftarrow{R} \{0,1\},\\ (\mathcal{X}_0, \mathcal{X}_1, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{pk}_{\mathsf{acc}}), (\mathsf{acc}_{\mathcal{X}_b}, \mathsf{aux}) \leftarrow \mathsf{Eval}((\mathsf{sk}^{\sim}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}),\\ \mathcal{X}_b), \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{t}}, \mathcal{O}^{\mathsf{u}}\},\ b^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}_b}, \mathsf{state}) : b = b^*\end{array}\right] \leq \frac{1}{2} + \epsilon(\kappa),$$

where $\mathcal{X}_0$ *and* $\mathcal{X}_1$ *are two distinct subsets of the accumulation domain and* $\mathcal{O}^{\mathsf{t}}$ *as well as* $\mathcal{O}^{\mathsf{u}}$ *are defined as follows:*

$$\mathcal{O}^{\mathsf{t}} := \begin{cases} \{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)}\} & \textit{if } \mathsf{t} = \mathsf{static},\\ \{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)}, \mathcal{O}^{\mathsf{A}_{\not\cup}(\cdot,\cdot,\mathsf{aux},\cdot)}, \mathcal{O}^{\mathsf{D}_{\cap}(\cdot,\cdot,\mathsf{aux},\cdot)}\} & \textit{otherwise.}\end{cases}$$

$$\mathcal{O}^{\mathsf{u}} := \begin{cases} \{\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\mathsf{aux},\cdot)}, \mathcal{O}^{\underline{\mathsf{W}}(\cdot,\cdot,\mathsf{aux},\cdot)}\} & \textit{if } \mathsf{u} = \mathsf{universal},\\ \{\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\mathsf{aux},\cdot)}\} & \textit{otherwise.}\end{cases}$$

If the probability above is exactly $1/2$ *we have* unconditional *indistinguishability, whereas we have* computational *indistinguishability if the probability is negligibly different from* $1/2$*.*

[3] Independently from our work, this observation was quite recently also made in [18] by the authors of [17]: The insertion of the random value has been removed from the game and the Eval algorithm is now required to be non-deterministic.

Here, $\mathcal{O}^{\mathsf{E}}$ is defined as before and all other oracles can only be called for the challenge accumulator. We require that the input parameter aux for the oracles is kept up to date and is provided by the environment, since the knowledge of aux would allow the adversary to trivially win the game. Furthermore, note that this game does not allow the adversary to control the randomness used for the evaluation of $\mathsf{acc}_{\mathcal{X}_b}$ (while it can be controlled when calling $\mathcal{O}^{\mathsf{E}}$). For the definitions of the remaining oracles, we use $\mathcal{X}_{\cup} := \mathcal{X}_0 \cup \mathcal{X}_1$ and $\mathcal{X}_{\cap} := \mathcal{X}_0 \cap \mathcal{X}_1$ to restrict the adversary from oracle queries which would trivially allow to win the game. $\mathcal{O}^{\mathsf{A}_{\not\cup}}$ as well as $\mathcal{O}^{\mathsf{D}_{\cap}}$ allow the adversary to execute the Add and Delete algorithms. Thereby, $\mathcal{O}^{\mathsf{A}_{\not\cup}}$ allows only queries for values $x_i \notin \mathcal{X}_{\cup}$, whereas $\mathcal{O}^{\mathsf{D}_{\cap}}$ allows only queries for values $x_i \in \mathcal{X}_{\cap}$. Furthermore, upon every Add and Delete the sets $\mathcal{X}_{\cup}$ and $\mathcal{X}_{\cap}$ are updated consistently. Oracles $\mathcal{O}^{\mathsf{W}}$ and $\mathcal{O}^{\underline{\mathsf{W}}}$ are as above, with the difference that $\mathcal{O}^{\mathsf{W}}$ allows only queries for values $x_i \in \mathcal{X}_{\cap}$, while $\mathcal{O}^{\underline{\mathsf{W}}}$ allows only queries for values $y_j \notin \mathcal{X}_{\cup}$.

Transformation 1. *On input a set $\mathcal{X}$, the* Eval *algorithm samples an element $x_r \notin \mathcal{X}$ uniformly at random from the accumulation domain. Next, it computes and returns* $(\mathsf{acc}_{\mathcal{X}'}, \mathsf{aux}')$ *for* $\mathcal{X}' \leftarrow \mathcal{X} \cup \{x_r\}$ *and* $\mathsf{aux}' \leftarrow (\mathsf{aux}, x_r)$.

Note that aux needs to be kept consistent for all other algorithms that require this input parameter. As already noted above, collision freeness no longer holds for $\mathcal{X}$ but with respect to $\mathcal{X} \cup \{x_r\}$. To draw a line between inherently randomized constructions and such relying on Transformation 1, we differentiate between indistinguishability and collision-freeness-weakening (cfw) indistinguishability:

Definition 9 (Indistinguishability). *Let $\mathcal{X}$ be the set in* $\mathsf{acc}_{\mathcal{X}_b}$. *A cryptographic accumulator is called* indistinguishable *if it is a secure, indistinguishable accumulator and* $\mathcal{X} = \mathcal{X}_b$.

Definition 10 (cfw-Indistinguishability). *Let $\mathcal{X}$ be the set in* $\mathsf{acc}_{\mathcal{X}_b}$. *A cryptographic accumulator is called* collision-freeness-weakening (cfw) indistinguishable *if it is a secure, indistinguishable accumulator and* $\mathcal{X} \neq \mathcal{X}_b$.

3.3 Relation Between Security Properties

Intuitively, undeniability seems to be a strictly stronger security requirement than collision freeness. We confirm this intuition below:

Lemma 1. *Every undeniable universal accumulator is collision-free.*

We prove the lemma above in the extended version of this paper.

As mentioned in [25], a black-box reduction in the other direction is impossible. [8] provides a collision-free universal accumulator that is not undeniable. Therefore, this proves the following lemma by counterexample:

Lemma 2. *Not every collision-free universal accumulator is undeniable.*

4 Categorizing Cryptographic Accumulators

Now, we give a comprehensive overview of existing accumulator schemes in Table 1. We categorize them regarding their static or dynamic nature and universal features and provide a characterization of their public updating capabilities (of witnesses and of accumulators, respectively). In particular, we tag an accumulator as dynamic, if witness and accumulator value updates can be performed in constant time, i.e., independent of the size of $\mathcal{X}$. If the same is possible without having access to the accumulator trapdoor, then we tag the accumulator as *publicly updatable*. Furthermore, the properties undeniability and indistinguishability have not been considered for most existing accumulator schemes so far. Therefore, we provide a classification regarding their indistinguishability (when using Transformation 1) and provide the respective proofs in the extended version. Likewise, we prove the undeniability of [3,16] in the extended version. For the sake of completeness, our comparison also includes static accumulator schemes [4,5,30,31].

5 Commitments from Indistinguishable Accumulators

In [14], it has been shown that universal dynamic accumulators can be black-box constructed from vector commitments. The question arises whether it is also possible to provide black-box constructions for certain types of commitments from indistinguishable accumulators. It is apparent that it is not possible to build vector commitments solely from accumulators in a black-box fashion, since their position binding would at least require some additional encoding. Nevertheless, we will show how to construct non-interactive commitments from indistinguishable 1-bounded accumulators. In the extended version, we show that such accumulators actually exist, i.e., we build the first indistinguishable t-bounded dynamic accumulator by modifying [29].

5.1 Black-Box Construction of Non-Interactive Commitments

Before we can start, we present a standard formal definition of non-interactive commitment schemes.

Definition 11 (Non-Interactive Commitment Scheme). *A non-interactive commitment scheme is a triple of efficient algorithms* $(\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open})$*, which are defined as follows:*

$\mathsf{Gen}(1^\kappa)$: *This (probabilistic) algorithm takes input a security parameter* κ *and outputs the public parameters* pp*.*

$\mathsf{Commit}(\mathsf{pp}, m)$: *This (probabilistic) algorithm takes input* pp *and a message* m *and outputs a commitment* C *together with a corresponding opening information* O*.*

$\mathsf{Open}(\mathsf{pp}, C, O)$: *This deterministic algorithm takes input* pp*, a commitment* C *with corresponding opening information* O *and outputs* $\perp$ *if* C *is not a valid commitment to any message and message* m *otherwise.*

Table 1. Overview of features of existing accumulator schemes. *Legend: D... dynamic, U... universal, Pub. Updates... constant cost for public updates of witnesses and accumulators, a... add, d... delete, ZK... zero-knowledge (non-)membership proofs, B... bounded,* $|\mathsf{pk_{acc}}|$ *... public parameter size,* $|\mathsf{wit}|$ *... membership witness size,* $|\underline{\mathsf{wit}}|$ *... non-membership witness size, Und... undeniability, Ind...* (cfw) *indistinguishability,* ✓ *... yes,* × *... no, - ... not available, ?... left open,* ‡ *... proven in the extended version of this paper.*

		Type		**Pub. Updates**										
						acc								
	Scheme	D	U	wit	$\underline{\mathsf{wit}}$	a	d	**ZK**	**B**	$\lvert\mathsf{pk_{acc}}\rvert$	$\lvert\mathsf{wit}\rvert$	$\lvert\underline{\mathsf{wit}}\rvert$	**Und.**	**Ind.**
s-RSA	BDM [5]	-	-	-	-	-	-	-	-	$\mathcal{O}(1)$	$\mathcal{O}(1)$	-	-	$\mathsf{cfw}^{‡}$
	BP [4]	-	-	-	-	-	-	-						$\mathsf{cfw}^{[17]}$
	CL [12]	✓	-	✓	-	✓	-	✓						$\mathsf{cfw}^{‡}$
	LLX [24]	✓	✓	✓	✓	✓	-	✓				$\mathcal{O}(1)$	?	$\times^{‡}$
t-SDH	NY [29]	✓	-	✓	-	-	-	✓	✓	$\mathcal{O}(t)$	$\mathcal{O}(1)$	-	-	$\mathsf{cfw}^{‡}$
	DT [16], ATSM [3]	✓	✓	✓	✓	-	-					$\mathcal{O}(1)$	✓‡	$\times^{‡}$
	This paper (extended version)	✓	-	✓	-	-	-			$\mathcal{O}(1)$		-	-	✓‡
t-DHE	CKS [11]	✓	-	✓	-	-	✓	✓	✓	$\mathcal{O}(t)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	-	$\times^{‡}$
VC	CF (RSA$^{∓}$, CDH†) [14]	✓	✓	✓	✓	✓	✓	-	✓	$\mathcal{O}(t)^{∓}$, $\mathcal{O}(t^2)^{†}$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	?	$\times^{‡}$
ZKS	This paper (Sec. 6)	-	✓	-	-	-	-	instantiation-dependent					✓‡	✓‡
CRH	BLL [7,8]	-	✓	-	-	-	-	-	-	$\mathcal{O}(1)$	$\mathcal{O}(\log t)$	$\mathcal{O}(\log t)$	✓	$\times^{‡}$
	CHKO [10]	-	✓	-	-	-	-	-	-				?	$\times^{‡}$
	BC [6]	-	-	-	-	-	-	✓	-				-	$\times^{‡}$ $\times^{[17]}$
ROM	NB [30,31]	-	-	-	-	-	-	-	✓	$\mathcal{O}(1)$	-	-	-	

For security, a non-interactive commitment scheme is required to provide *correctness*, *binding* and *hiding*. We omit a formal definition of correctness as it is straightforward. The remaining properties are defined as follows.

Definition 12 (Binding). *A non-interactive commitment scheme is* binding*, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{Gen}(1^\kappa), (C^*, O^*, O'^*) \leftarrow \mathcal{A}(\mathsf{pp}), m \leftarrow \mathsf{Open}(\mathsf{pp}, C^*, O^*), \\ m' \leftarrow \mathsf{Open}(\mathsf{pp}, C^*, O'^*) : m \neq m' \quad \wedge \quad m \neq \bot \quad \wedge \quad m' \neq \bot\end{array}\right] \leq \epsilon(\kappa).$$

Definition 13 (Hiding). *A non-interactive commitment scheme is* hiding*, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{Gen}(1^\kappa), (m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{pp}), b \xleftarrow{R} \{0,1\}, \\ (C, O) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_b), b^* \leftarrow \mathcal{A}(\mathsf{pp}, C, \mathsf{state}) : \\ b = b^*\end{array}\right] \leq \frac{1}{2} + \epsilon(\kappa).$$

In Scheme 1, we present a black-box construction of commitments from indistinguishable accumulators and prove the so obtained construction secure (Theorem 1). Before we continue, we want to recall that in the trusted setup model all algorithms can be correctly executed without $\mathsf{sk}_{\mathsf{acc}}$.

$\mathsf{Gen}(1^\kappa)$: This algorithm runs $(\mathsf{sk}_{\widetilde{\mathsf{acc}}}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Acc.Gen}(1^\kappa, 1)$, discards $\mathsf{sk}_{\mathsf{acc}}$ and returns $\mathsf{pp} \leftarrow \mathsf{pk}_{\mathsf{acc}}$.

$\mathsf{Commit}(\mathsf{pp}, m)$: This algorithm chooses randomness r, runs $(C, \mathsf{aux}) \leftarrow \mathsf{Eval}_r((\emptyset, \mathsf{pk}_{\mathsf{acc}}), m)$, computes $\mathsf{wit}_m \leftarrow \mathsf{WitCreate}((\emptyset, \mathsf{pk}_{\mathsf{acc}}), C, \mathsf{aux}, m)$, sets $O \leftarrow (r, m, \mathsf{wit}_m, \mathsf{aux})$ and returns (C, O).

$\mathsf{Open}(\mathsf{pp}, C, O)$: This algorithm checks whether $\mathsf{Eval}_r((\emptyset, \mathsf{pk}_{\mathsf{acc}}), m) \stackrel{?}{=} C$ and whether $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, C, \mathsf{wit}_m, m) \stackrel{?}{=} \mathsf{true}$ and returns m on success and $\bot$ otherwise.

Scheme 1: Commitment Scheme from Indistinguishable Accumulators

Theorem 1. *If indistinguishable* 1*-bounded accumulators exist, then non-interactive commitments exist as well.*

We prove Theorem 1 in the extended version of this paper.

The black-box construction from Scheme 1 can easily be extended to support commitments to sets (where the opening is always with respect to the entire set) by setting the bound t of the bounded accumulator to the desired set size. Furthermore, using $\mathsf{sk}_{\mathsf{acc}}$ as trapdoor, one can also construct trapdoor commitments.

We finally note that cfw-indistinguishable accumulators (and hence also Transformation 1) are not useful for constructing commitments. The reason for this is that the accumulation of the additional random value immediately breaks the binding property.

6 Zero-Knowledge Sets Imply Indistinguishable Undeniable Accumulators

Zero-knowledge sets (ZK-sets) [27] allow to commit to a set $\mathcal{X}$ and then prove predicates of the form $x_i \in \mathcal{X}$ or $x_i \notin \mathcal{X}$ without revealing anything else about the set. We observe that ZK-sets can be used to model indistinguishable, unbounded, undeniable accumulators. Unfortunately, there is no formal security definition for zero-knowledge sets (in [23] only the algorithms are formalized, while security is stated informally). However, zero-knowledge sets are a special instance of zero-knowledge elementary databases (ZK-EDB) [27]. ZK-EDBs store key-value pairs and when querying the database with a key, the respective value is returned (or $\perp$ if the given key is not contained in the EDB). Thereby, no further information about the remaining EDB leaks. Therefore, ZK-sets are ZK-EDBs where the values for all contained keys are set to 1 (or the values are omitted at all). We can, thus, define the security on the basis of the models in [15,27] as follows.

Definition 14 (ZK-set). *A ZK-set is a tuple of efficient algorithms* $(\mathsf{Gen}, \mathsf{Commit}, \mathsf{Query}, \mathsf{Verify})$, *which are defined as follows:*

$\mathsf{Gen}(1^\kappa)$: *This (probabilistic) algorithm takes input a security parameter κ and outputs a public key* pk.

$\mathsf{Commit}(\mathsf{pk}, \mathcal{X})$: *This algorithm takes input the public key* pk *and a set $\mathcal{X}$ and outputs a commitment C to $\mathcal{X}$.*

$\mathsf{Query}(\mathsf{pk}, \mathcal{X}, C, x)$: *This algorithm takes input the public key* pk, *a set $\mathcal{X}$, a corresponding commitment C and and value x. It outputs a proof π_x if $x \in \mathcal{X}$ and a proof $\underline{\pi}_x$ if $x \notin \mathcal{X}$.*

$\mathsf{Verify}(\mathsf{pk}, C, x, \pi_x/\underline{\pi}_x)$: *This algorithm takes input the public key* pk, *a commitment C and a value x. Furthermore, it either takes a membership proof π_x or a non-membership proof $\underline{\pi}_x$ (denoted by $\pi_x/\underline{\pi}_x$). It outputs* `true` *if the proof can be correctly verified and* `false` *otherwise.*

For security, ZK-sets require *perfect completeness*, *soundness* and *zero-knowledge.* Perfect completeness requires that for every honestly generated key, every honestly computed commitment C, value x and corresponding proof $\pi_x/\underline{\pi}_x$, the Verify algorithm always returns `true`. Since this property is straightforward, we do not formally state it here. We formally define the remaining properties:

Definition 15 (Soundness). *A ZK-set is* sound, *if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{c}\mathsf{pk} \leftarrow \mathsf{Gen}(1^\kappa), (C^*, x^*, \pi_x^*, \underline{\pi}_x^*) \leftarrow \mathcal{A}(\mathsf{pk}) : \\ \mathsf{Verify}(\mathsf{pk}, C^*, \pi_x^*, x^*) = \mathtt{true} \quad \wedge \quad \mathsf{Verify}(\mathsf{pk}, C^*, \underline{\pi}_x^*, x^*) = \mathtt{true}\end{array}\right] \leq \epsilon(\kappa)$$

Definition 16 (Zero Knowledge). *A ZK-set is* zero-knowledge*, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

$$\left| \Pr\left[\begin{array}{c} \mathsf{pk} \leftarrow \mathsf{Gen}(1^\kappa), \\ (\mathcal{X}, \mathsf{state}_\mathcal{A}) \leftarrow \mathcal{A}(\mathsf{pk}), \\ C \leftarrow \mathsf{Commit}(\mathsf{pk}, \mathcal{X}), \\ \mathcal{A}^{\mathcal{O}^\mathsf{Q}(\cdot,\mathcal{X},\cdot,\cdot)}(\mathsf{state}_\mathcal{A}, \\ \mathsf{pk}, C) = \mathtt{true} \end{array}\right] - \Pr\left[\begin{array}{c} (\mathsf{pk}, \mathsf{state}_\mathcal{S}) \leftarrow \mathcal{S}^\mathsf{G}(1^\kappa), \\ (\mathcal{X}, \mathsf{state}_\mathcal{A}) \leftarrow \mathcal{A}(\mathsf{pk}), \\ (C, \mathsf{state}'_\mathcal{S}) \leftarrow \mathcal{S}^\mathsf{E}(\mathsf{pk}, \mathsf{state}_\mathcal{S}), \\ \mathcal{A}^{\mathcal{S}^\mathsf{Q}(\mathsf{state}'_\mathcal{S},\cdot,\mathcal{X},\cdot,\cdot)}(\mathsf{state}_\mathcal{A}, \\ \mathsf{pk}, C) = \mathtt{true} \end{array}\right] \right| \leq \epsilon(\kappa)$$

Here, $\mathcal{O}^\mathsf{Q}$ allows the adversary to execute the Query algorithm, whereas $\mathcal{S} = (\mathcal{S}^\mathsf{G}, \mathcal{S}^\mathsf{E}, \mathcal{S}^\mathsf{Q})$ denotes a PPT simulator, which allows to execute the simulated Gen, Eval and Query algorithms, respectively. We note that the definition above is tailored to cover computational zero-knowledge. It could, however, easily be modified to also cover statistical or perfect zero knowledge.

In Scheme 2 we present a black-box construction of indistinguishable unbounded undeniable accumulators from ZK-sets.

$\mathsf{Gen}(1^\kappa)$**:** This algorithm runs $\mathsf{pk} \leftarrow \mathsf{ZKS.Gen}(1^\kappa)$ and returns $(\mathsf{sk}_\mathsf{acc}, \mathsf{pk}_\mathsf{acc}) \leftarrow (\emptyset, \mathsf{pk})$.

$\mathsf{Eval}((\emptyset, \mathsf{pk}_\mathsf{acc}), \mathcal{X})$**:** This algorithm runs $\mathsf{acc}_\mathcal{X} \leftarrow \mathsf{ZKS.Commit}(\mathsf{pk}_\mathsf{acc}, \mathcal{X})$ and returns $\mathsf{acc}_\mathcal{X}$ together with $\mathsf{aux} \leftarrow \mathcal{X}$.

$\mathsf{WitCreate}((\emptyset, \mathsf{pk}_\mathsf{acc}),\ \mathsf{acc}_\mathcal{X},\ \mathsf{aux},\ x_i,\ \mathsf{type})$**:** This algorithm obtains $\mathcal{X}$ from aux and runs $\pi_{x_i}/\underline{\pi_{x_i}} \leftarrow \mathsf{ZKS.Query}(\mathsf{pk}, \mathcal{X}, \mathsf{acc}_\mathcal{X}, x_i)$. If $\pi_{x_i}/\underline{\pi}_{x_i}$ conflicts with the requested witness type, it returns $\perp$. Otherwise it returns $\mathsf{wit}_{x_i} \leftarrow \pi_{x_i}$ or $\underline{\mathsf{wit}}_{x_i} \leftarrow \underline{\pi}_{x_i}$, respectively.

$\mathsf{Verify}(\mathsf{pk}_\mathsf{acc},\ \mathsf{acc},\ \mathsf{wit}_{x_i},\ x_i,\ \mathsf{type})$**:** This algorithm checks whether type conflicts with the type of the supplied witness and returns $\perp$ if so. Otherwise it returns the result of $\mathsf{ZKS.Verify}(\mathsf{pk}, \mathsf{acc}, x_i, \mathsf{wit}_{x_i})$.

Scheme 2: Indistinguishable Unbounded Undeniable Accumulator from ZK-Sets

Theorem 2. *If ZK-sets exist, then indistinguishable, unbounded, undeniable accumulators exist as well.*

We prove Theorem 2 in the extended version of this paper.

The above black-box construction yields the first construction of indistinguishable undeniable accumulators. We note that it is, however, questionable whether the two notions of ZK-sets and indistinguishable undeniable accumulators are equivalent (as the simulation based model of zero-knowledge appears to be stronger than the game based indistinguishability model).

In [23], Kate et al. introduced nearly ZK-sets. The difference to ordinary ZK-sets is that nearly ZK-sets have a public upper bound on the cardinality of set $\mathcal{X}$. It is apparent that these constructions imply indistinguishable t-bounded undeniable accumulators. In further consequence, this means that nearly ZK-sets can also be used to construct commitments (cf. Section 5).

References

1. 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 (2011)
2. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 1–20. Springer, Heidelberg (2012)
3. 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 (2009)
4. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)
5. Benaloh, J., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
6. Boneh, D., Corrigan-Gibbs, H.: Bivariate polynomials modulo composites and their applications. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 42–62. Springer, Heidelberg (2014). http://eprint.iacr.org/2014/719
7. Buldas, A., Laud, P., Lipmaa, H.: Accountable certificate management using undeniable attestations. In: ACM CCS, pp. 9–17. ACM (2000)
8. Buldas, A., Laud, P., Lipmaa, H.: Eliminating Counterevidence with Applications to Accountable Certificate Management. Journal of Computer Security **10** (2002)
9. Camacho, P., Hevia, A.: On the impossibility of batch update for cryptographic accumulators. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 178–188. Springer, Heidelberg (2010)
10. Camacho, P., Hevia, A., Kiwi, M., 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 (2008)
11. 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 (2009)
12. Camenisch, J.L., 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 (2002)
13. Canard, S., Jambert, A.: On extended sanitizable signature schemes. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 179–194. Springer, Heidelberg (2010)
14. 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 (2013)
15. Chase, M., Healy, A., Lysyanskaya, A., Malkin, T., Reyzin, L.: Mercurial Commitments with Applications to Zero-Knowledge Sets. Journal of Cryptology **26**(2), 251–279 (2013)
16. Damgård, I., Triandopoulos, N.: Supporting Non-membership Proofs with Bilinear-map Accumulators. Cryptology ePrint Archive, Report 2008/538 (2008). http://eprint.iacr.org/2008/538
17. de Meer, H., Liedel, M., Pöhls, H.C., Posegga, J.: Indistinguishability of One-Way Accumulators. Technical Report MIP-1210, Faculty of Computer Science and Mathematics (FIM), University of Passau (2012)

18. de Meer, H., Pöhls, H.C., Posegga, J., Samelin, K.: Redactable signature schemes for trees with signer-controlled non-leaf-redactions. In: Obaidat, M.S., Filipe, J. (eds.) ICETE 2012. CCIS, vol. 455, pp. 155–171. Springer, Heidelberg (2014)
19. Fauzi, P., Lipmaa, H., Zhang, B.: Efficient non-interactive zero knowledge arguments for set operations. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 214–231. Springer, Heidelberg (2014). http://eprint.iacr.org/2014/006
20. Fazio, N., Nicolisi, A.: Cryptographic Accumulators: Definitions. Constructions and Applications, Technical report (2002)
21. Ghosh, E., Ohrimenko, O., Tamassia, R.: Verifiable Member and Order Queries on a List in Zero-Knowledge. Cryptology ePrint Archive, Report 2014/632 (2014). http://eprint.iacr.org/2014/632
22. Goodrich, M.T., Tamassia, R., Hasic, J.: An efficient dynamic and distributed cryptographic accumulator. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 372–388. Springer, Heidelberg (2002)
23. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010)
24. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007)
25. Lipmaa, H.: Secure accumulators from euclidean rings without trusted setup. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 224–240. Springer, Heidelberg (2012)
26. Mashatan, A., Vaudenay, S.: A Fully Dynamic Universal Accumulator. Proceedings of the Romanian Academy **14**, 269–285 (2013)
27. Micali, S., Rabin, M.O., Kilian, J.: Zero-knowledge sets. In: FOCS, pp. 80–91 (2003)
28. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed E-cash from bitcoin. In: IEEE Symposium on Security and Privacy, pp. 397–411. IEEE (2013)
29. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
30. Nyberg, K.: Commutativity in cryptography. In: 1st International Trier Conference in Functional Analysis. Walter Gruyter & Co (1996)
31. Nyberg, K.: Fast accumulated hashing. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 83–87. Springer, Heidelberg (1996)
32. Peng, K., Bao, F.: Vulnerability of a non-membership proof scheme. In: SECRYPT, pp. 1–4, July 2010
33. Pöhls, H.C., Peters, S., Samelin, K., Posegga, J., de Meer, H.: Malleable signatures for resource constrained platforms. In: Cavallaro, L., Gollmann, D. (eds.) WISTP 2013. LNCS, vol. 7886, pp. 18–33. Springer, Heidelberg (2013)
34. Pöhls, H.C., Samelin, K.: On updatable redactable signatures. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 457–475. Springer, Heidelberg (2014)
35. Sander, T.: Efficient accumulators without trapdoor extended abstract. In: Varadharajan, V., Mu, Y. (eds.) ICICS 1999. LNCS, vol. 1726, pp. 252–262. Springer, Heidelberg (1999)
36. Sander, T., Ta-Shma, A., Yung, M.: Blind, auditable membership proofs. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 53–71. Springer, Heidelberg (2001)

37. Slamanig, D.: Dynamic accumulator based discretionary access control for outsourced storage with unlinkable access. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 215–222. Springer, Heidelberg (2012)
38. Sudarsono, A., Nakanishi, T., Funabiki, N.: Efficient proofs of attributes in pairing-based anonymous credential system. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 246–263. Springer, Heidelberg (2011)
39. 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 (2003)
40. Wang, P., Wang, H., Pieprzyk, J.: A new dynamic accumulator for batch updates. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 98–112. Springer, Heidelberg (2007)