# 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 $\mathsf{wit}_{x_i}$ to certify 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

* The authors have been supported by the European Commission through project FP7-FutureID, grant agreement number 318424. This is the full version of a paper to appear at CT-RSA 2015.

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 [8] and authenticated dictionaries [24] have been proposed. Accumulators are also used as building block in redactable [35, 36], sanitizable [14], $P$-homomorphic signatures [2], anonymous credentials [40], group signatures [41], privacy-preserving data outsourcing [39] as well as for authenticated data structures [22]. 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 [38]) are now widely used for revocation of group signatures and anonymous credentials [13]. Quite recently, accumulators were also used in Zerocoin [30], 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 [37], Sander proposed to use RSA moduli with unknown factorization to construct trapdoor-free accumulators. Camenisch and Lysyanskaya [13] 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. [26] added support for non-membership witnesses to [13] 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 [28, 34]. Lipmaa [27] 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 [41], Tsudik and Xu proposed a variation of [13], 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 [42] an accumulator scheme that allows to accumulate arbitrary integers and supports batch updates of witnesses has been proposed. Yet, this scheme was broken in [10].

**Known order groups:** In [31], 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 [17] 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. [12]. It supports public updates of witnesses and the accumulator and its security relies on the $t$-DHE assumption.

**Hash-based constructions:** Buldas et al. [8, 9] presented the very first universal dynamic accumulator that satisfies *undeniability* (termed as undeniable attester and formalized in context of accumulators in [27]). 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 [11] (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 [7].

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 [32, 33].

**Contribution:** The contributions of this paper are as follows:

- While some papers [3–5, 13, 31] do not explicitly formalize accumulator schemes, formal definitions are given in [1, 11, 12, 15, 21, 26, 27, 42]. 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 [8, 9, 27] and a strengthened version of the recent indistinguishability notion [18]. 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 [20, 25], commitments to vectors [15] and to zero-knowledge sets [29], 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 $\mathsf{acc}$ 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 $\mathsf{wit}_{x_i}$, 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 $a \xleftarrow{R} A$, we denote that $a$ is chosen uniformly at random from the set $A$.

A function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ 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 $\epsilon$ to denote such a negligible function.

**Definition 1 (Bilinear map:).** *A bilinear map (pairing) is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ being cyclic groups of prime order $p$. Let $g_1$ and $g_2$ generate $\mathbb{G}_1$ and $\mathbb{G}_2$. We require $e$ to be efficiently computable and that the following conditions hold:*

***Bilinearity:*** $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a) \quad \forall a, b \in \mathbb{Z}_p$

***Non-degeneracy:*** $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, *i.e.,* $e(g_1, g_2)$ *generates* $\mathbb{G}_T$.

If $\mathbb{G}_1 = \mathbb{G}_2$ we talk about *symmetric* pairings, whereas we talk about *asymmetric* pairings otherwise.

**Definition 2 (Strong RSA assumption (s-RSA) [4]).** *Given two appropriately chosen primes $p$ and $q$ such that $N = pq$ has bitlength $\kappa$ and a random $u \in \mathbb{Z}_N^*$, then it holds for all PPT adversaries $\mathcal{A}$ that*

$$\mathsf{Pr}\left[(v, w) \leftarrow \mathcal{A}(u, N) : v^w \equiv u \pmod{N}\right] \leq \epsilon(\kappa).$$

**Definition 3 ($t$-SDH assumption [6]).** *Let $p$ be a prime of bitlength $\kappa$, $\mathbb{G}$ be a finite cyclic group of order $p$, $g$ be a generator of $\mathbb{G}$, $\alpha \in_R \mathbb{Z}_p^*$ and $t > 0$. Then, for all PPT adversaries $\mathcal{A}$ it holds that*

$$\mathsf{Pr}\left[\left(c, g^{\frac{1}{\alpha + c}}\right) \leftarrow \mathcal{A}(g, g^{\alpha}, \ldots, g^{\alpha^t})\right] \leq \epsilon(\kappa) \quad \text{for some } c \in \mathbb{Z}_p \setminus \{-\alpha\}.$$

**Definition 4 ($t$-DHE assumption [12, 23]).** *Let $p$ be a prime of bitlength $\kappa$, $\mathbb{G}$ be a finite cyclic group of order $p$, $g$ be a generator of $\mathbb{G}$ and $\gamma \in_R \mathbb{Z}_p^*$. Then, for all PPT adversaries $\mathcal{A}$ it holds that*

$$\mathsf{Pr}\left[g^{\gamma^{t+1}} \leftarrow \mathcal{A}(g, g^{\gamma^1}, \ldots, g^{\gamma^t}, g^{\gamma^{t+2}}, \ldots, g^{\gamma^{2t}})\right] \leq \epsilon(\kappa).$$

## 3 A Unified Model for Cryptographic Accumulators

In the original sense, accumulator schemes were defined by the following properties (see, e.g., [13, 26]). 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 $\kappa$, defines a functionality $f : \mathcal{Z}_A \times \mathcal{Z}_I \to \mathcal{Z}_A$, 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 $f(f(\mathsf{acc}, x_1), x_2) = f(f(\mathsf{acc}, x_2), x_1) \quad \forall x_1, x_2 \in \mathcal{Z}_I, \mathsf{acc} \in \mathcal{Z}_\mathsf{A}$.

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 $\mathsf{Gen}$ 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 [27] proposed a modified model which divides the $\mathsf{Gen}$ algorithm into a $\mathsf{Setup}$ and a $\mathsf{Gen}$ algorithm. In this model, the adversary can control the randomness used inside $\mathsf{Setup}$ and, thus, knows the trapdoor. Nevertheless, it can neither access nor influence the randomness of the $\mathsf{Gen}$ 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 [21, 42]. 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 [12, 15], 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.

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

[1] This model is tailored to the hidden order group setting, where $\mathsf{Setup}$ produces a composite modulus $N$. $\mathsf{Gen}$ 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.

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 $\mathsf{sk}_{\mathsf{acc}}^{\sim}$) 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 $\mathsf{sk}_{\mathsf{acc}}^{\sim}$ is set, the party running the algorithm needs to be fully trusted.

**Definition 5 (Static Accumulator).** *A static accumulator is a tuple of efficient algorithms* (Gen, Eval, WitCreate, Verify) *which are defined as follows:*

$\mathsf{Gen}(1^{\kappa}, t)$**:** *This algorithm takes a security parameter $\kappa$ 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.*

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

$\mathsf{WitCreate}((\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$**:** *This algorithm takes a key pair* $(\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}})$*, 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* $\mathsf{wit}_{x_i}$ *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* $\mathsf{wit}_{x_i}$ *and a value $x_i$. It returns* `true` *if* $\mathsf{wit}_{x_i}$ *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 5, we can now formalize *dynamic accumulators*. We widely align our definitions with [21,42], 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 6 (Dynamic Accumulator).** *A dynamic accumulator is a static accumulator that additionally provides efficient algorithms* (Add, Delete, WitUpdate) *which are defined as follows:*

$\mathsf{Add}((\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$**:** *This algorithm takes a key pair* $(\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}})$*, 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′*.*

$\mathsf{Delete}((\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$**:** *This algorithm takes a key pair* $(\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}})$*, 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′*.*

$\mathsf{WitUpdate}((\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{wit}_{x_i}, \mathsf{aux}, x_j)$**:** *This algorithm takes a key pair* $(\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}})$*, a witness* $\mathsf{wit}_{x_i}$ *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* $\mathsf{wit}'_{x_i}$ *on success and $\perp$ otherwise.*

[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 $\mathsf{Eval}_r$.

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 7 (Universal Accumulator).** *A universal accumulator is a static or a dynamic accumulator with the following properties. For static accumulator schemes the algorithms* 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* WitUpdate*.*

### 3.2 Security Model

Now, we introduce a security model for accumulators, which we adapt from [26] 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 $\mathsf{sk_{acc}}$. 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 8 (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:*

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

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

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

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

Thereby, $\mathcal{O}^{\mathsf{E}}, \mathcal{O}^{\mathsf{A}}$ and $\mathcal{O}^{\mathsf{D}}$ represent the oracles for the algorithms Eval, 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 9 (Secure Accumulator).** *A cryptographic accumulator is* secure *if it is correct and collision-free.*

**Undeniable accumulators:** In [27], 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 10 (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:*

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

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

**Indistinguishable accumulators:** Li et al. [26] 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. [18] 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 [26] weakens collision resistance. Basically, one can easily generate a membership witness for the random value. Secondly, the security game in [18] 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 $\mathsf{Eval}$ 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 $\mathsf{Eval}$ algorithm without modifying the set $\mathcal{X}$ (as, for instance, done in Section 5.1). Apparently, the latter option depends on the specific accumulator scheme, whereas the shortcomings in [18] can be addressed by introducing a generic transformation for the former approach (cf. Transformation 1).

**Definition 12 (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:*

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

*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:*

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

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

*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.*

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 $\mathsf{aux}$ for the oracles is kept up to date and is provided by the environment, since the knowledge of $\mathsf{aux}$ 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. $\mathcal{O}^{\mathsf{A}_{\not\in}}$ as well as $\mathcal{O}^{\mathsf{D}_\cap}$ allow the adversary to execute the $\mathsf{Add}$ and $\mathsf{Delete}$ algorithms. Thereby, $\mathcal{O}^{\mathsf{A}_{\not\in}}$ 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 $\mathsf{Add}$ and $\mathsf{Delete}$ 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 $y_j \notin \mathcal{X}_\cup$.

[3] Independently from our work, this observation was quite recently also made in [19] by the authors of [18]: The insertion of the random value has been removed from the game and the $\mathsf{Eval}$ algorithm is now required to be non-deterministic.

**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 $\mathsf{aux}$ 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 ($\mathsf{cfw}$) indistinguishability:

**Definition 13 (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 14 ($\mathsf{cfw}$-Indistinguishability).** *Let $\mathcal{X}$ be the set in $\mathsf{acc}_{\mathcal{X}_b}$. A cryptographic accumulator is called* collision-freeness-weakening ($\mathsf{cfw}$) 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 Appendix C.1.

As mentioned in [27], a black-box reduction in the other direction is impossible. [9] 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 State of the Art and Categorization

In this section we discuss the basic principles of existing constructions grouped by their underlying security assumptions. Subsequently, we provide a compact overview and exhaustive classification of existing approaches.

### 4.1 Strong RSA Setting

All schemes in this setting are extensions of [4,5]. Here, the accumulator $\mathsf{acc}_{\mathcal{X}}$ is defined to be $\mathsf{acc}_{\mathcal{X}} \leftarrow g^{\prod_{x \in \mathcal{X}} x} \bmod N$, where $N$ is an RSA modulus consisting of two large safe primes $p, q$ and $g$ is randomly drawn from the cyclic group of quadratic residues modulo $N$. Thus, we have $(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}) = ((p, q), N)$ and a witness for a value $x_i$ is given by $\mathsf{wit}_{x_i} \leftarrow \mathsf{acc}_{\mathcal{X}}^{x_i^{-1}} \bmod N$. Clearly, if we were able to forge a witness $\mathsf{wit}_{x_i} \equiv \mathsf{acc}_{\mathcal{X}}^{x_i^{-1}} \pmod N$ for a value $x_i$ not contained in $\mathsf{acc}$, then we would also be able to break the strong RSA assumption. Due to the multiplicative relationship of the accumulated values in the exponent, the domain of accumulated values is restricted to prime numbers (or products of primes with unknown factorization [41]). Note that accumulating a

composite number $a = b \cdot c$ would allow to derive witnesses for each of its factors, when given a witness $\mathsf{wit}_a$ for $a$ (i.e., $\mathsf{wit}_b \equiv (\mathsf{wit}_a)^c \pmod N$). Hence, to accumulate sets from more general domains, one needs a suitable mapping from these domains to prime numbers (e.g., [38]).

Some accumulator schemes in this setting [13, 26] also provide dynamic features. Adding a value to the accumulator can be done without any secret by a simple exponentiation of the accumulator and its witnesses. In contrast, if one wants to delete a value $x_j$, then one has to compute the $x_j$-th root of the accumulator, which is intractable without $\mathsf{sk}_{\mathsf{acc}}$ under the strong RSA assumption. Yet, one can still use an arithmetic trick to publicly update membership witnesses upon the deletion of a value. To update the witness for a value $x_i$ in $\mathsf{acc}_{\mathcal{X}\setminus\{x_j\}}$, one finds $a, b \in \mathbb{Z}$ such that $ax_i + bx_j = 1$ and computes the new witness as $\mathsf{wit}'_{x_i} \leftarrow {\mathsf{wit}_{x_i}}^b \cdot {\mathsf{acc}_{\mathcal{X}\setminus\{x_j\}}}^a \bmod N$ from the old witness $\mathsf{wit}_{x_i}$ [13].

Furthermore, the accumulator scheme in [26] also provides universal features as it supports non-membership witnesses: Let $\mathsf{acc}_{\mathcal{X}}$ be an accumulator to the set $\mathcal{X}$ and let $y_j \notin \mathcal{X}$. Now, it holds that $\gcd(\prod_{x\in\mathcal{X}} x, y_j) = 1$ or, equivalently, $a\prod_{x\in\mathcal{X}} x + by_j = 1$ for $a, b \in \mathbb{Z}$. Consequently, we compute $d \leftarrow g^{-b} \bmod N$, where $g$ is the initial value of the empty accumulator, and form a non-membership witness $\underline{\mathsf{wit}}_{y_j} \leftarrow (a, d)$. The verification of non-membership witnesses is, then, done by checking whether ${\mathsf{acc}_{\mathcal{X}}}^a \equiv d^{y_j} \cdot g \pmod N$ holds. In a similar way as it is done for membership witnesses, also non-membership witnesses can be updated publicly (cf. [26]).

### 4.2 *t*-SDH Setting

All schemes in this settings are based on the $t$-bounded accumulator proposed by Nguyen [31], which uses a group $\mathbb{G}$ of prime order $p$ generated by $g$ with a bilinear map $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$. Here, we have $\mathsf{pk}_{\mathsf{acc}} = (g, g^s, g^{s^2}, \ldots, g^{s^t}, u)$ and $\mathsf{sk}_{\mathsf{acc}} = s$. An accumulator $\mathsf{acc}_{\mathcal{X}}$ to a set $\mathcal{X} = \{x_1, \ldots, x_n\} \subseteq \mathbb{Z}_p$ with $n \leq t$ is defined to be $\mathsf{acc}_{\mathcal{X}} \leftarrow g^{u\prod_{x\in\mathcal{X}}(x+s)}$ and a membership witness for a value $x_i \in \mathcal{X}$ is computed as $\mathsf{wit}_{x_i} \leftarrow g^{u\prod_{x\in\mathcal{X}\setminus\{x_i\}}(x+s)}$, where $u \xleftarrow{R} \mathbb{Z}_p^*$. Then, one checks whether a value $x_i$ is contained in $\mathsf{acc}_{\mathcal{X}}$ by verifying whether $e(\mathsf{acc}_{\mathcal{X}}, g) = e(g^{x_i}g^s, \mathsf{wit}_{x_i})$ holds. This scheme allows to evaluate accumulators publicly, that is, by expanding the polynomial $h(X) = \prod_{x\in\mathcal{X}}(x + X) \in \mathbb{Z}_p[X]$ and evaluating it in $\mathbb{G}$ via $\mathsf{pk}_{\mathsf{acc}}$, which results in $g^{h(s)}$. The public computation of a witness for $x_i$ works likewise with regard to the set $\mathcal{X} \setminus \{x_i\}$. Furthermore, these witnesses can also be updated in constant time and without knowing the secret key (cf. [31]).

In [3, 17], Nguyen's scheme is extended by non-membership witnesses and the random value $u$ is eliminated. The former work also shows how public updates of non-membership witnesses can be done in constant time. Note that these tweaks can also be applied to the latter. The computation of a non-membership witness for a value $y_j \notin \mathcal{X}$ in [3] exploits the fact that the polynomial division of $h(X) = \prod_{x\in\mathcal{X}}(x + X)$ by $(y_j + X)$ leaves a remainder $d \in \mathbb{Z}_p^*$. Such a witness has the form $(a, d) = (g^{\frac{h(s)-d}{y_j+s}}, d)$ and can be verified via $e(\mathsf{acc}_{\mathcal{X}}, g) \stackrel{?}{=} e(a, g^{y_j}g^s)e(g, g^d)$. In [17], $d$ is set to $h(-y_j)$, which also yields suitable non-membership witnesses.

### 4.3 $t$-DHE Setting

In [12], Camenisch et. al give a $t$-bounded accumulator scheme based on the $t$-DHE assumption. Like the accumulators in the $t$-SDH setting, it uses a group $\mathbb{G}$ of prime order $p$ generated by $g$ with a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. In addition, it requires a signature scheme with a corresponding keypair $(\mathsf{sk}_{\mathsf{sig}}, \mathsf{pk}_{\mathsf{sig}})$. Here, we have $\mathsf{sk}_{\mathsf{acc}} = \mathsf{sk}_{\mathsf{sig}}$ and the public key is $\mathsf{pk}_{\mathsf{acc}} = (g_1, \ldots, g_t, g_{t+2}, \ldots, g_{2t}, z, \mathsf{pk}_{\mathsf{sig}}) = (g^{\gamma^1}, \ldots, g^{\gamma^t}, g^{\gamma^{t+2}}, \ldots, g^{\gamma^{2t}}, e(g,g)^{\gamma^{t+1}}, \mathsf{pk}_{\mathsf{sig}})$ with $\gamma \xleftarrow{R} \mathbb{Z}_p^*$. We can accumulate a set $\mathcal{X} = \{x_1, \ldots, x_m\}$ with $m \leq t$ by computing $\mathsf{acc}_{\mathcal{X}} \leftarrow \prod_{i=1}^{m} g_{t+1-i}$ and signing $g_i$ together with $x_i$ using $\mathsf{sk}_{\mathsf{sig}}$, which assigns the value of $x_i$ to $g_i$. A witness $\mathsf{wit}_{x_j}$ for $x_j \in \mathcal{X}$ is given as $\mathsf{wit}_{x_j} \leftarrow \prod_{i=1, i\neq j}^{m} g_{t+1-i+j}$. The membership of $x_j$ can be verified by checking whether $e(g_j, \mathsf{acc}_{\mathcal{X}}) = z \cdot e(g, \mathsf{wit}_{x_j})$ holds and by verifying the signature on $g_j$ and $x_j$ under $\mathsf{pk}_{\mathsf{sig}}$.

This scheme enables public updates of the witnesses and the accumulator upon delete, since this requires only $\mathsf{pk}_{\mathsf{acc}}$. If we, however, want to add a value $x_i$ to the accumulator, then we need the secret signing key $\mathsf{sk}_{\mathsf{acc}}$ to create a signature on $g_i$ and $x_i$ in order to link value $x_i$ with parameter $g_i$. Consequently, public additions to the accumulator require to include a signature for every potential value to be accumulated into the public parameters. Obviously, with the exception of very small accumulation domains this seems impractical.

### 4.4 Collision-Resistant Hash Setting

In this setting, accumulators are built from collision-resistant hash functions and (sorted) Merkle hash-trees. Here, each leaf node represents an accumulated value and is labeled with the corresponding hash value. Every inner node is labeled with a hash value formed from its children's labels (and potentially some additional information). The accumulator $\mathsf{acc}$ itself is the root node label (root hash) and a membership witness for an accumulated value $x_i$ is its authentication path, i.e., all the labels of siblings on the path of the leaf to the root node. Verification of witnesses is done the obvious way, i.e., by recomputing the hash tree and comparing the root hashes. To prove non-membership, such schemes exploit the order on the leaf nodes [8, 9, 11] (alternatively, an order can also be enforced by a suitable encoding on the accumulated values [11]). Proving non-membership of a value $y_j$ boils down to demonstrating membership witnesses for two values $x_i, x_{i+1}$ corresponding to two consecutive leaves such that $x_i < y_j < x_{i+1}$ holds.

### 4.5 Accumulators from Vector Commitments

Catalano and Fiore [15] proposed a black-box construction of accumulators from vector commitments. A vector commitment allows to form a succinct commitment $C$ to a vector $\mathcal{X} = (x_1, \ldots, x_n)$. Here, it should be computationally infeasible to open position $i$ of $C$ to a value $x_i'$ different from $x_i$. The accumulation domain in the black-box construction in [15] is the set $D = \{1, \ldots, t\}$. The accumulator is modeled as a commitment to a binary vector of length $t$, that is, each bit $i$ indicates the presence or absence of element $i \in D$ in the accumulator. Then, the (non-)membership of a value $i$ can be proven by opening position $i$ of a commitment to 1 or 0, respectively.

### 4.6 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 Appendix B. Likewise, we prove the undeniability of [3, 17] in Appendix B. For the sake of completeness, our comparison also includes static accumulator schemes [4, 5, 32, 33].

## 5 Commitments from Indistinguishable Accumulators

In [15], 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. We start by proposing the first indistinguishable $t$-bounded accumulator construction (for arbitrary $t$) and, then, provide the black-box construction based on any such scheme.

### 5.1 An Indistinguishable $t$-Bounded Dynamic Accumulator

Here, we will build an indistinguishable $t$-bounded accumulator from the $t$-SDH based dynamic accumulator in [31]. This construction already uses a randomizer (denoted as $u$) that is chosen by $\mathsf{Gen}$ and added to $\mathsf{pk}_{\mathsf{acc}}$. As a consequence, the $\mathsf{Eval}$ algorithm is still deterministic. In order to obtain indistinguishability, we modify the way $u$ is used and observe that randomly choosing it on each call to $\mathsf{Eval}$ yields an indistinguishable accumulator. In Appendix A, we present a modified version of [31] incorporating the aforementioned functionality and prove the following theorem.

**Theorem 1.** *Under the t-SDH assumption, Scheme 3 (cf. Appendix A) is an indistinguishable t-bounded dynamic accumulator.*

### 5.2 Black-Box Construction of Non-Interactive Commitments

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

**Definition 15 (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:*

| | Scheme | Type D | Type U | Pub. Updates wit | Pub. Updates $\underline{\mathsf{wit}}$ | Pub. Updates acc a | Pub. Updates acc d | ZK | B | $\lvert\mathsf{pk}_{\mathsf{acc}}\rvert$ | $\lvert\mathsf{wit}\rvert$ | $\lvert\underline{\mathsf{wit}}\rvert$ | Und. | Ind. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| s-RSA | BDM [5] | - | - | - | - | - | - | - | - | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | - | - | $\mathsf{cfw}^{\ddagger}$ |
| s-RSA | BP [4] | - | - | - | - | - | - | - | - | | | | | $\mathsf{cfw}$ [18] |
| s-RSA | CL [13] | ✓ | - | ✓ | - | ✓ | - | ✓ | - | | | | | $\mathsf{cfw}^{\ddagger}$ |
| s-RSA | LLX [26] | ✓ | ✓ | ✓ | ✓ | ✓ | - | ✓ | - | | | $\mathcal{O}(1)$ | ? | $\times^{\ddagger}$ |
| t-SDH | NY [31] | ✓ | - | ✓ | - | - | - | ✓ | ✓ | $\mathcal{O}(t)$ | $\mathcal{O}(1)$ | - | - | $\mathsf{cfw}^{\ddagger}$ |
| t-SDH | DT [17], ATSM [3] | ✓ | ✓ | ✓ | ✓ | - | - | | | | | $\mathcal{O}(1)$ | ✓‡ | $\times^{\ddagger}$ |
| t-SDH | This paper (Sec. 5.1) | ✓ | - | ✓ | - | - | - | | | $\mathcal{O}(1)$ | | - | - | ✓‡ |
| t-DHE | CKS [12] | ✓ | - | ✓ | - | - | ✓ | ✓ | ✓ | $\mathcal{O}(t)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | - | $\times^{\ddagger}$ |
| VC | CF (RSA$^{\mp}$, CDH$^{\dagger}$) [15] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | - | ✓ | $\mathcal{O}(t)^{\mp}$, $\mathcal{O}(t^2)^{\dagger}$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | ? | $\times^{\ddagger}$ |
| ZKS | This paper (Sec. 6) | - | ✓ | - | - | - | - | instantiation-dependent | | | | | ✓‡ | ✓‡ |
| CRH | BLL [8,9] | - | ✓ | - | - | - | - | - | - | $\mathcal{O}(1)$ | $\mathcal{O}(\log t)$ | $\mathcal{O}(\log t)$ | ✓ | $\times^{\ddagger}$ |
| CRH | CHKO [11] | - | ✓ | - | - | - | - | - | - | | | | ? | $\times^{\ddagger}$ |
| CRH | BC [7] | - | - | - | - | - | - | ✓ | - | | | | - | $\times^{\ddagger}$ |
| ROM | NB [32,33] | - | - | - | - | - | - | - | ✓ | $\mathcal{O}(1)$ | - | - | - | $\times$ [18] |

**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,* $\lvert\mathsf{pk}_{\mathsf{acc}}\rvert$ *... public parameter size,* $\lvert\mathsf{wit}\rvert$ *... membership witness size,* $\lvert\underline{\mathsf{wit}}\rvert$ *... non-membership witness size, Und... undeniability, Ind...* ($\mathsf{cfw}$) *indistinguishability,* ✓ *... yes,* × *... no, - ... not available, ?... left open,* $^{\ddagger}$ *... proven in this paper.*

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

$\mathsf{Commit}(\mathsf{pp}, m)$: *This (probabilistic) algorithm takes input* $\mathsf{pp}$ *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* $\mathsf{pp}$*, 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.*

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

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

**Definition 17 (Hiding).** *A non-interactive commitment scheme is* hiding*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that*

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

In Scheme 1, we present a black-box construction of commitments from indistinguishable accumulators and prove the so obtained construction secure (Theorem 2). 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 $(\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow \mathsf{Acc.Gen}(1^\kappa, 1)$, 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 $\perp$ otherwise.

**Scheme 1:** Commitment Scheme from Indistinguishable Accumulators

**Theorem 2.** *If indistinguishable* 1*-bounded accumulators exist, then non-interactive commitments exist as well.*

We prove Theorem 2 in Appendix C.2.

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) [29] 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 [25] 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) [29]. 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 [16, 29] as follows.

**Definition 18 (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* $\kappa$ *and outputs a public key* $\mathsf{pk}$*.*

$\mathsf{Commit}(\mathsf{pk}, \mathcal{X})$: *This algorithm takes input the public key* $\mathsf{pk}$ *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* $\mathsf{pk}$*, a set* $\mathcal{X}$*, a corresponding commitment* $C$ *and and value* $x$*. It outputs a proof* $\pi_x$ *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* $\mathsf{pk}$*, a commitment* $C$ *and a value* $x$*. Furthermore, it either takes a membership proof* $\pi_x$ *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 $\mathsf{Verify}$ algorithm always returns `true`. Since this property is straightforward, we do not formally state it here. We formally define the remaining properties:

**Definition 19 (Soundness).** *A ZK-set is* sound*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that*

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

**Definition 20 (Zero Knowledge).** *A ZK-set is* zero-knowledge*, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that*

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

Here, $\mathcal{O}^{\mathsf{Q}}$ allows the adversary to execute the $\mathsf{Query}$ 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 $\mathsf{Gen}$, $\mathsf{Eval}$ and $\mathsf{Query}$ 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 $\mathsf{aux}$ 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 $\mathsf{type}$, 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 $\mathsf{type}$ 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 3.** *If ZK-sets exist, then indistinguishable, unbounded, undeniable accumulators exist as well.*

We prove Theorem 3 in Appendix C.3.

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 [25], 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. Tolga Acar and Lan Nguyen. Revocation for Delegatable Anonymous Credentials. In *PKC*, volume 6571 of *LNCS*, pages 423–440. Springer, 2011.

2. Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on Authenticated Data. In *TCC*, volume 7194 of *LNCS*, pages 1–20. Springer, 2012.
3. Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. Dynamic Universal Accumulators for DDH Groups and Their Application to Attribute-Based Anonymous Credential Systems. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *LNCS*, pages 295–308. Springer, 2009.
4. Niko Baric and Birgit Pfitzmann. Collision-free Accumulators and Fail-stop Signature Schemes Without Trees. In *EUROCRYPT*, pages 480–494, 1997.
5. Josh Benaloh and Michael de Mare. One-way Accumulators: A Decentralized Alternative to Digital Signatures. In *EUROCRYPT*, pages 274–285, 1993.
6. Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. In *EUROCRYPT*, pages 56–73, 2004.
7. Dan Boneh and Henry Corrigan-Gibbs. Bivariate Polynomials Modulo Composites and their Applications. In *ASIACRYPT*, 2014. http://eprint.iacr.org/2014/719.
8. Ahto Buldas, Peeter Laud, and Helger Lipmaa. Accountable Certificate Management Using Undeniable Attestations. In *ACM CCS*, pages 9–17. ACM, 2000.
9. Ahto Buldas, Peeter Laud, and Helger Lipmaa. Eliminating Counterevidence with Applications to Accountable Certificate Management. *Journal of Computer Security*, 10:2002, 2002.
10. Philippe Camacho and Alejandro Hevia. On the Impossibility of Batch Update for Cryptographic Accumulators. In Michel Abdalla and PauloS.L.M. Barreto, editors, *LATINCRYPT*, volume 6212 of *LNCS*, pages 178–188. Springer Berlin Heidelberg, 2010.
11. Philippe Camacho, Alejandro Hevia, Marcos A. Kiwi, and Roberto Opazo. Strong Accumulators from Collision-Resistant Hashing. In *ISC*, volume 5222 of *LNCS*, pages 471–486. Springer, 2008.
12. Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In *PKC*, Irvine, pages 481–500, Berlin, Heidelberg, 2009. Springer-Verlag.
13. Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *CRYPTO*, pages 61–76, 2002.
14. Sébastien Canard and Amandine Jambert. On Extended Sanitizable Signature Schemes. In *CT-RSA*, volume 5985 of *LNCS*, pages 179–194. Springer, 2010.
15. Dario Catalano and Dario Fiore. Vector Commitments and Their Applications. In *PKC*, volume 7778 of *LNCS*, pages 55–72, 2013. Extended version: http://eprint.iacr.org/2011/495.
16. Melissa Chase, Alexander Healy, Anna Lysyanskaya, Tal Malkin, and Leonid Reyzin. Mercurial Commitments with Applications to Zero-Knowledge Sets. *Journal of Cryptology*, 26(2):251–279, 2013.
17. Ivan Damgård and Nikos Triandopoulos. Supporting Non-membership Proofs with Bilinear-map Accumulators. Cryptology ePrint Archive, Report 2008/538, 2008. http://eprint.iacr.org/2008/538.
18. Hermann de Meer, Manuel Liedel, Henrich C. Pöhls, and Joachim Posegga. Indistinguishability of One-Way Accumulators. Technical Report MIP-1210, Faculty of Computer Science and Mathematics (FIM), University of Passau, 2012.
19. Hermann de Meer, Henrich C. Pöhls, Joachim Posegga, and Kai Samelin. Redactable Signature Schemes for Trees with Signer-Controlled Non-Leaf-Redactions. In Mohammad S. Obaidat and Joaquim Filipe, editors, *E-Business and Telecommunications*, volume 455 of *CCIS*, pages 155–171. Springer, 2014.
20. Prastudy Fauzi, Helger Lipmaa, and Bingsheng Zhang. Efficient Non-Interactive Zero Knowledge Arguments for Set Operations. In *FC*, LNCS, 2014. http://eprint.iacr.org/2014/006.

21. Nelly Fazio and Antonio Nicolisi. Cryptographic Accumulators: Definitions, Constructions and Applications. Technical report, 2002.
22. Esha Ghosh, Olga Ohrimenko, and Roberto Tamassia. Verifiable Member and Order Queries on a List in Zero-Knowledge. Cryptology ePrint Archive, Report 2014/632, 2014. http://eprint.iacr.org/2014/632.
23. Philippe Golle, Stanislaw Jarecki, and Ilya Mironov. Cryptographic Primitives Enforcing Communication and Storage Complexity. In *FC*, volume 2357 of *LNCS*, pages 120–135. Springer, 2003.
24. Michael T. Goodrich, Roberto Tamassia, and Jasminka Hasic. An Efficient Dynamic and Distributed Cryptographic Accumulator. In *ISC*, volume 2433 of *LNCS*, pages 372–388. Springer, 2002.
25. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 177–194. Springer, 2010.
26. Jiangtao Li, Ninghui Li, and Rui Xue. Universal Accumulators with Efficient Nonmembership Proofs. In *ACNS*, volume 4521 of *LNCS*, pages 253–269. Springer, 2007.
27. Helger Lipmaa. Secure Accumulators from Euclidean Rings without Trusted Setup. In *ACNS*, volume 7341 of *LNCS*, pages 224–240, 2012.
28. Atefeh Mashatan and Serge Vaudenay. A Fully Dynamic Universal Accumulator. *Proceedings of the Romanian Academy*, 14:269–285, 2013.
29. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-Knowledge Sets. In *FOCS*, pages 80–91, 2003.
30. Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *IEEE Symposium on Security and Privacy*, pages 397–411. IEEE, 2013.
31. Lan Nguyen. Accumulators from Bilinear Pairings and Applications. In *CT-RSA*, LNCS, pages 275–292. Springer, 2005.
32. Kaisa Nyberg. Commutativity in Cryptography. In *1st International Trier Conference in Functional Analysis*. Walter Gruyter & Co, 1996.
33. Kaisa Nyberg. Fast accumulated hashing. In Dieter Gollmann, editor, *FSE*, volume 1039 of *LNCS*, pages 83–87. Springer, 1996.
34. Kun Peng and Feng Bao. Vulnerability of a non-membership proof scheme. In *SECRYPT*, pages 1–4, July 2010.
35. Henrich C. Pöhls, Stefan Peters, Kai Samelin, Joachim Posegga, and Hermann de Meer. Malleable Signatures for Resource Constrained Platforms. In *WISTP*, volume 7886 of *LNCS*, pages 18–33. Springer, 2013.
36. Henrich C. Pöhls and Kai Samelin. On Updatable Redactable Signatures. In *ACNS*, volume 8479 of *LNCS*. Springer, 2014.
37. Tomas Sander. Efficient Accumulators Without Trapdoor. In *ICICS*, pages 252–262. Springer, 1999.
38. Tomas Sander, Amnon Ta-Shma, and Moti Yung. Blind, Auditable Membership Proofs. In *FC*, LNCS, pages 53–71. Springer, 2000.
39. Daniel Slamanig. Dynamic Accumulator based Discretionary Access Control for Outsourced Storage with Unlinkable Access. In *FC*, LNCS. Springer, 2012.
40. Amang Sudarsono, Toru Nakanishi, and Nobuo Funabiki. Efficient Proofs of Attributes in Pairing-Based Anonymous Credential System. In *PETS*, pages 246–263, 2011.
41. Gene Tsudik and Shouhuai Xu. Accumulating Composites and Improved Group Signing. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *LNCS*, pages 269–286. Springer, 2003.
42. Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. A New Dynamic Accumulator for Batch Updates. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS*, volume 4861 of *LNCS*, pages 98–112. 2007.

# A An Indistinguishable $t$-Bounded Dynamic Accumulator

Scheme 3 presents a modified version of [31]. Subsequently, we prove it to be a secure indistinguishable $t$-bounded dynamic accumulator. For sake of consistency, the value $u$ used in [31] will be denoted by $r$.

$\mathsf{Gen}(1^\kappa, t)$**:** On input security parameter $\kappa$, this algorithm chooses two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order $p$ such that $\log_2 p = \kappa$ having a pairing $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$. Additionally, it chooses a generator $g$ of $\mathbb{G}$, as well as $s \xleftarrow{R} \mathbb{Z}_p^*$. Finally, it returns $(\mathsf{sk}_{\mathsf{acc}}, \mathsf{pk}_{\mathsf{acc}}) \leftarrow (s, (g^{s^i})_{i=0}^t)$.

$\mathsf{Eval}_r((\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}}), \mathcal{X})$**:** On input a key pair $(\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}})$ and $\mathcal{X} = \{x_1, \ldots, x_n\}$, this algorithm chooses $r \xleftarrow{R} \mathbb{Z}_p^*$. If $\mathsf{sk}_{\mathsf{acc}} \neq \emptyset$, it computes $\mathsf{acc}_{\mathcal{X}} \leftarrow g^{r\prod_{i=1}^n (x_i+s)}$. Otherwise, it expands the polynomial $\prod_{x\in\mathcal{X}}(x + X)$ to $\sum_{i=0}^n a_i \cdot X^i$ and computes $\mathsf{acc}_{\mathcal{X}} \leftarrow (\prod_{i=0}^n (g^{s^i})^{a_i})^r$. Finally, it returns $\mathsf{acc}_{\mathcal{X}}$ and $\mathsf{aux} \leftarrow (r, \mathcal{X})$.

$\mathsf{WitCreate}((\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$**:** On input a key pair $(\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}})$, $\mathsf{acc}_{\mathcal{X}}$, $\mathsf{aux} = (r, \mathcal{X})$ and $x_i$, this algorithm checks whether $x_i \in \mathcal{X}$ and returns $\perp$ otherwise. If $\mathsf{sk}_{\mathsf{acc}} \neq \emptyset$, it computes and returns $\mathsf{wit}_{x_i} \leftarrow \mathsf{acc}_{\mathcal{X}}^{(x_i+s)^{-1}}$. Otherwise, it expands the polynomial $\prod_{x\in\mathcal{X}\setminus\{x_i\}}(x + X)$ to $\sum_{i=0}^{n-1} a_i \cdot X^i$ where $\mathcal{X} = \{x_1, \ldots, x_n\}$ and returns $\mathsf{wit}_{x_i} \leftarrow \left(\prod_{i=0}^{n-1}(g^{s^i})^{a_i}\right)^r$.

$\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{acc}_{\mathcal{X}}, \mathsf{wit}_{x_i}, x_i)$**:** This algorithm checks whether $e(\mathsf{acc}_{\mathcal{X}}, g) = e(\mathsf{wit}_{x_i}, g^{x_i} g^s)$ holds. If so, it returns `true` and `false` otherwise.

$\mathsf{Add}((\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$**:** On input a key pair $(\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}})$, $\mathsf{acc}_{\mathcal{X}}$, $\mathsf{aux} = (r, \mathcal{X})$ and $x_i$, this algorithm checks whether $x_i \notin \mathcal{X}$ and returns $\perp$ otherwise. If $\mathsf{sk}_{\mathsf{acc}} \neq \emptyset$ it computes $\mathsf{acc}' \leftarrow \mathsf{acc}_{\mathcal{X}}^{(x_i+s)}$. Otherwise, with $\mathcal{X} = \{x_1, \ldots, x_n\}$, it expands the polynomial $\prod_{x\in\mathcal{X}\cup\{x_i\}}(x + X)$ to $\sum_{i=0}^{n+1} a_i \cdot X^i$ and computes $\mathsf{acc}' \leftarrow \left(\prod_{i=0}^{n+1}(g^{s^i})^{a_i}\right)^r$. Finally, it returns $\mathsf{acc}'$ and $\mathsf{aux}' \leftarrow \mathcal{X} \cup \{x_i\}$.

$\mathsf{Delete}((\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, \mathsf{aux}, x_i)$**:** On input a key pair $(\mathsf{sk}_{\mathsf{acc}}^{\sim}, \mathsf{pk}_{\mathsf{acc}})$, $\mathsf{acc}_{\mathcal{X}}$, $\mathsf{aux} = (r, \mathcal{X})$ and $x_i$, this algorithm checks whether $x_i \in \mathcal{X}$ or $\mathcal{X} \neq \emptyset$ and returns $\perp$ otherwise. If $\mathsf{sk}_{\mathsf{acc}} \neq \emptyset$ this algorithm computes $\mathsf{acc}' \leftarrow \mathsf{acc}_{\mathcal{X}}^{(x_i+s)^{-1}}$. Otherwise, it expands the polynomial $\prod_{x\in\mathcal{X}\setminus\{x_i\}}(x + X)$ to $\sum_{i=0}^{n-1} a_i \cdot X^i$ where $\mathcal{X} = \{x_1, \ldots, x_n\}$ and computes $\mathsf{acc}' \leftarrow \left(\prod_{i=0}^{n-1}(g^{s^i})^{a_i}\right)^r$. Finally, it returns $\mathsf{acc}'$ and $\mathsf{aux}' \leftarrow \mathcal{X} \setminus \{x_i\}$.

$\mathsf{WitUpdate}(\mathsf{pk}_{\mathsf{acc}}, \mathsf{wit}_{x_i}, \mathsf{aux}, x_j)$**:** This algorithm obtains as input a public key $\mathsf{pk}_{\mathsf{acc}}$, $\mathsf{wit}_{x_i}$, $\mathsf{aux} = (\mathsf{acc}', \mathsf{acc}, x_i, \mathsf{op})$ and $x_j$. Here, $\mathsf{acc}'$ represents the already updated accumulator and $\mathsf{acc}$ the accumulator before the update. Information $\mathsf{op}$ determines whether $x_j$ was deleted or added to the accumulator $\mathsf{acc}'$. It returns the updated witness $\mathsf{wit}'_{x_i} \leftarrow \mathsf{acc} \cdot \mathsf{wit}_{x_i}^{x_j - x_i}$ if $\mathsf{op} = 0$ or $\mathsf{wit}'_{x_i} \leftarrow (\mathsf{acc}'^{-1} \cdot \mathsf{wit}_{x_i})^{\frac{1}{x_j - x_i}}$ if $\mathsf{op} = 1$.

**Scheme 3:** Indistinguishable $t$-Bounded Dynamic Accumulator Scheme

## A.1 Security Proofs of the Modified Scheme

The correctness proofs in [3, 31] also apply to our scheme, and, hence, we do not prove it explicitly here.

*Proof (Collision freeness).* Similar to [3], we prove the collision freeness for membership witnesses by showing that an efficient adversary $\mathcal{A}$ against the collision freeness can be

turned into an efficient adversary $\mathcal{B}$ against the $t$-SDH assumption. $\mathcal{B}$ gets a $t$-SDH instance $(g, g^s, \ldots, g^{s^t})$ as input and starts the game by setting $\mathsf{pk}_{\mathsf{acc}} = (g, g^s, \ldots, g^{s^t})$ and handing $\mathsf{pk}_{\mathsf{acc}}$ over to $\mathcal{A}$. Moreover, $\mathcal{B}$ keeps a mapping $M$ of accumulators $\mathsf{acc}_{\mathcal{X}}$ to their accumulated sets with corresponding randomizers, which is consistently updated with respect to the oracle queries. $\mathcal{B}$ simulates the oracles for $\mathcal{A}$ as follows:

- $\mathcal{O}^{\mathsf{E}}$**:** For a given $\mathsf{pk}_{\mathsf{acc}}$, set $\mathcal{X} = \{x_1, \ldots, x_n\}$ and randomizer $r$, call $(\mathsf{acc}_{\mathcal{X}}, \mathsf{aux}) \leftarrow \mathsf{Eval}_r((\emptyset, \mathsf{pk}_{\mathsf{acc}}), \mathcal{X})$, add $(\mathsf{acc}_{\mathcal{X}}, \mathcal{X}, r)$ to $M$ and return $(\mathsf{acc}_{\mathcal{X}}, \mathsf{aux})$.
- $\mathcal{O}^{\mathsf{A}}$**:** For a given $\mathsf{pk}_{\mathsf{acc}}$, accumulator $\mathsf{acc}_{\mathcal{X}}$ and element $x_i$ to be added, lookup the set $U = \{u_1, \ldots, u_n\}$ and randomizer $r$ corresponding to $\mathsf{acc}_{\mathcal{X}}$ in $M$ and return $\perp$ if $x_i \in U$. We assume that $U = \emptyset$ and $r \xleftarrow{R} \mathbb{Z}_p$ when $\mathsf{acc}_{\mathcal{X}}$ is not contained in $M$. Compute $(\mathsf{acc}'_{\mathcal{X}}, \mathsf{aux}') \leftarrow \mathsf{Add}((\emptyset, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, (r, U), x_i)$, add $(\mathsf{acc}_{\mathcal{X}}, U \cup \{x_i\}, r)$ to $M$ and return $\mathsf{acc}'_{\mathcal{X}}$ and $\mathsf{aux}'$.
- $\mathcal{O}^{\mathsf{D}}$**:** For a given $\mathsf{pk}_{\mathsf{acc}}$, accumulator $\mathsf{acc}_{\mathcal{X}}$ and element $x_i$ to be deleted, lookup the set $U = \{u_1, \ldots, u_n\}$ and randomizer $r$ corresponding to $\mathsf{acc}_{\mathcal{X}}$ in $M$ and return $\perp$ if $x_i \notin U$ or $\mathsf{acc}_{\mathcal{X}}$ is not contained in $M$. Compute $(\mathsf{acc}'_{\mathcal{X}}, \mathsf{aux}') \leftarrow \mathsf{Delete}((\emptyset, \mathsf{pk}_{\mathsf{acc}}), \mathsf{acc}_{\mathcal{X}}, (r, U), x_i)$, add $(\mathsf{acc}_{\mathcal{X}}, U \setminus \{x_i\}, r)$ to $M$ and return $\mathsf{acc}'_{\mathcal{X}}$ and $\mathsf{aux}'$.
- $\mathcal{O}^{\mathsf{W}}$**:** For a given $\mathsf{pk}_{\mathsf{acc}}$, accumulator $\mathsf{acc}_{\mathcal{X}}$, and element $x_i$, lookup the set $U = \{u_1, \ldots, u_n\}$ and randomizer $r$ corresponding to $\mathsf{acc}_{\mathcal{X}}$ in $M$ and return $\perp$ if $x_i \notin U$. Otherwise, return $\mathsf{wit}_{x_i} \leftarrow \mathsf{WitCreate}((\emptyset, \mathsf{pk}_{\mathsf{acc}}), (r, U), x_i)$.

Eventually, $\mathcal{A}$ outputs a membership witness $\mathsf{wit}^*_{x_j}$ for some value $x^*_j \notin \mathcal{X}^*$, a set $\mathcal{X}^*$ and a randomizer $r^*$ such that the verification relation $e(\mathsf{wit}^*_{x_j}, g^{x^*_j}\ g^s) = e(\mathsf{acc}_{\mathcal{X}^*}, g)$ holds, where $(\mathsf{acc}_{\mathcal{X}^*}, \mathsf{aux}) \leftarrow \mathsf{Eval}_r((\emptyset, \mathsf{pk}_{\mathsf{acc}}), \mathcal{X}^*)$. Then, $\mathcal{B}$ knows the polynomial $h(X) = \prod_{x \in \mathcal{X}^*}(x + X)$ , the polynomial $\phi(X)$ and $d$ such that $h(X) = \phi(X)(x_j + X) + d$ holds (because $x^*_j \notin \mathcal{X}^*$). Then, $\mathcal{B}$ can compute $g^{r^* \cdot \phi(s)}$ by expanding the polynomial $\phi(X)$ to $\sum_{i=0}^{|\mathcal{X}^*|-1} a_i \cdot X^i$ and computing $g^{r^* \cdot \phi(s)} \leftarrow (\prod_{i=0}^{|\mathcal{X}^*|-1}(g^{s^i})^{a_i})^{r^*}$. Thus, $\mathcal{B}$ can output

$$\left(\mathsf{wit}^*_{x_j} \cdot \left(g^{r^* \cdot \phi(s)}\right)^{-1}\right)^{\frac{1}{r^* \cdot d}} \leftarrow \left(g^{\frac{r^* \cdot h(s)}{x^*_j + s}}\ g^{-\frac{r^* \cdot (h(s)-d)}{x^*_j + s}}\right)^{\frac{1}{r^* \cdot d}} = \left(g^{\frac{r^* \cdot d}{x^*_j + s}}\right)^{\frac{1}{r^* \cdot d}} = g^{\frac{1}{x^*_j + s}}$$

together with $x^*_j$ as solution to the $t$-SDH problem. □

*Proof (Indistinguishability).* We recall that the indistinguishability game is defined such that it is not possible to obtain $g^r$ as $\mathcal{X}_0 \neq \mathcal{X}_1$ and only values from $x \in \mathcal{X}_0 \cap \mathcal{X}_1$ can be deleted via $\mathcal{O}^{\mathsf{D}}$. This means that it is impossible to extract the randomizer $r$ used upon $\mathsf{Eval}$. The adversary sees $\mathsf{acc}_{\mathcal{X}_b} = g^{r \prod_{x \in \mathcal{X}_b}(x+s)}$, and for values $x_j \in \mathcal{X}_0 \cap \mathcal{X}_1$ it is possible to obtain witnesses $\mathsf{wit}_{x_j} = g^{r \prod_{x \in \mathcal{X}_b \setminus \{x_j\}}(x+s)}$. Furthermore, it is possible to add values $x \notin \mathcal{X}_0 \cup \mathcal{X}_1$ to and to delete values $x \in \mathcal{X}_0 \cap \mathcal{X}_1$ from the challenge accumulator $\mathsf{acc}_{\mathcal{X}_b}$.

There exist randomizers $r_0$ and $r_1$ for both candidate sets $\mathcal{X}_0$ and $\mathcal{X}_1$, respectively, such that $\mathsf{acc}_{\mathcal{X}_b} = g^{r_0 \prod_{x \in \mathcal{X}_0}(x+s)} = g^{r_1 \prod_{x \in \mathcal{X}_1}(x+s)}$. Since it is not possible to obtain the randomness $r$ used in $\mathsf{Eval}$, even an unbounded adversary can not decide whether $r = r_0$ or $r = r_1$, meaning that it is not possible to distinguish whether $\mathcal{X}_0$ or $\mathcal{X}_1$ was accumulated.

Likewise, for all witnesses $\mathsf{wit}_{x_j}$, where $x_j \in \mathcal{X}_0 \cap \mathcal{X}_1$, there are representations consistent with the values $r_0$ and $r_1$ such that $\mathsf{wit}_{x_j} = g^{r_0 \prod_{x \in \mathcal{X}_0 \setminus \{x_j\}}(x+s)} = g^{r_1 \prod_{x \in \mathcal{X}_1 \setminus \{x_j\}}(x+s)}$, which means that witnesses do not give an additional advantage. The same argumentation also holds when adding/deleting values. □

## B Security of Existing Schemes in our Model

In the following, we will analyze the undeniability and (in)distinguishability of several constructions. We will not reconsider the collision freeness of these constructions in our model, as it is compatible with previous models.

### B.1 Undeniability of Universal Accumulators

We will now prove the subsequent lemma:

**Lemma 3.** *The universal accumulators in [3, 17] are undeniable.*

*Proof.* We prove the undeniability of [3, 17] by showing that an efficient adversary $\mathcal{A}$ against the undeniability can be turned into an efficient adversary $\mathcal{B}$ against $t$-SDH assumption. Note that $\mathcal{B}$ can simulate all oracles, since, due to the model, the accumulator and the witnesses can be generated without knowing the trapdoor $\mathsf{sk}_{\mathsf{acc}}$. $\mathcal{B}$ gets a $t$-SDH instance $(g, g^s, \ldots, g^{s^t})$ as input, sets $\mathsf{pk}_{\mathsf{acc}} = (g, g^s, \ldots, g^{s^t})$ and starts $\mathcal{A}$ with $\mathsf{pk}_{\mathsf{acc}}$.

Eventually, $\mathcal{A}$ outputs $(\mathsf{wit}^*_{x_i}, \underline{\mathsf{wit}}^*_{x_i}, x_i^*, \mathsf{acc}^*)$, with $\underline{\mathsf{wit}}^*_{x_i} = (a, d)$ such that $e(\mathsf{acc}^*, g) = e(g^{x_i^*} g^s, \mathsf{wit}^*_{x_i})$ and $e(\mathsf{acc}^*, g) = e(g^{x_i^*} g^s, a) e(g^d, g)$ holds. Then it holds that $e(g^{x_i^*} g^s, \mathsf{wit}^*_{x_i}) = e(g^{x_i^*} g^s, a) e(g^d, g)$. With, $\mathsf{wit}_{x_i} = g^{\mathcal{X}'}$ and $a = g^{a'}$ for some unknown $\mathcal{X}', a'$, we know that the relation of the exponents is $\mathcal{X}' = a' + \frac{d}{x_i^* + s}$. Thus, $\mathcal{B}$ can compute $g^{\frac{1}{x_i^* + s}} \leftarrow (\mathsf{wit}^*_{x_i} \cdot a^{-1})^{d^{-1}}$ and output it as a solution to the $t$-SDH problem. □

### B.2 Indistinguishability of (Dynamic) Accumulators

We investigate the indistinguishability of [3–5, 8, 9, 11, 13, 15, 17, 26, 31]. As we will see, most existing *dynamic* accumulator schemes can be turned into cfw-indistinguishable accumulators under Transformation 1, whereas this does not work for existing *universal* accumulators.

**Lemma 4.** *Under Transformation 1, the schemes in [4], [5] and [13] are* cfw*-indistinguishable.*

*Proof.* De Meer et al. [18] proved a version of [4], where Eval randomly chooses a private starting value $g$, to be indistinguishable in their model.[4] The random choice of $g$ is a way to apply Transformation 1. Thus, we can reuse the proof in [18] for cfw-indistinguishability.

The schemes in [4] and [5] are identical, with the only difference that the accumulation domain in [4] is restricted to prime numbers. Since the proof in [18] does not require this restriction, indistinguishability also holds for [5]. The difference from [13] to [4, 5] is that [13] supports dynamic updates. We, thus, need to show that access to $\mathcal{O}^{\mathsf{A}}$ and $\mathcal{O}^{\mathsf{D}}$ does not give the adversary any advantage. The proof in [18] shows that the value of the challenge accumulator is random, when $x$ is randomly sampled from the accumulation domain, i.e., it cannot be distinguished from a randomly sampled starting value of an empty accumulator. Taking this, together with the restriction that only elements

[4] We note that in their security game an additional random value $x$ is inserted into the accumulator. This value is, however, not required for their indistinguishability proof.

$a \notin \mathcal{X}_\cup$ can be added and only elements $a \in \mathcal{X}_\cap$ can be deleted, the adversary does not learn more than when dealing with an empty accumulator. Consequently, it can not win the indistinguishability game with a probability being non-negligibly greater than 1/2. The same argumentation also holds for [26] under the assumption that the *universal* features, i.e., the non-membership witnesses/proofs, are not being used. □

**Lemma 5.** *Under Transformation 1, the scheme in [31] is* cfw*-indistinguishable.*

When applying Transformation 1, the accumulator in [31] is computed as $\mathsf{acc}_\mathcal{X} \leftarrow g^{u(x+s)\prod_{i=1}^{n}(x_i+s)}$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$, $u$ is a random value in $\mathbb{Z}_p^*$ (chosen during $\mathsf{Gen}$) and $x$ is a random value from the accumulation domain (freshly chosen upon each call to $\mathsf{Eval}$). As discussed in Section 5.1, slight changes to the accumulator scheme in [31] make it indistinguishable. In the following, we show the cfw-indistinguishability under Transformation 1.

*Proof.* It is easy to see that for a given accumulator $\mathsf{acc}_\mathcal{X}$ to set $\mathcal{X} = \{x_1, \ldots, x_n\}$, that is, $\mathsf{acc}_\mathcal{X} = g^{u(x+s)\prod_{i=1}^{n}(x_i+s)}$ (after applying Transformation 1), another value $x'$ exists such that for an arbitrarily chosen set $W = \{w_i, \ldots w_m\}$ it holds that $\mathsf{acc}_\mathcal{X} = g^{u(x'+s)\prod_{i=1}^{m}(w_i+s)}$. More precisely, this means that the accumulated set is unconditionally hidden within $\mathsf{acc}_\mathcal{X}$. Moreover, since the witnesses have the same form as the accumulator, they also do not reveal anything beyond the fact that an element has indeed been accumulated. □

Note that it is crucial that the random elements, which are inserted into the accumulator, are not reused to reach this unconditional indistinguishability. Similarly, the schemes in [3, 17, 26], which extend [13] and [31] by universal features, are cfw-indistinguishable when solely used with membership witnesses. This is due to the fact that in these schemes non-membership witnesses leak information about the accumulated values (cf. Appendix B.3).

**Lemma 6.** *Under Transformation 1, the scheme in [12] is distinguishable.*

*Proof.* Here the accumulator $\mathsf{acc}$ is a product of generators contained in $\mathsf{pk}_\mathsf{acc}$. Thus, the size of $\mathsf{pk}_\mathsf{acc}$ is linear in the maximum number of accumulated elements. This means that one can efficiently brute force the generators contained in $\mathsf{acc}_{\mathcal{X}_b}$ and win the game if $\mathcal{X}_0$ and $\mathcal{X}_1$ are chosen such that $||\mathcal{X}_0| - |\mathcal{X}_1|| \geq 1$. □

**Lemma 7.** *Under Transformation 1, the scheme in [7] is distinguishable.*

Here, the accumulator is the root of a perfect Merkle hash-tree, where a statistically hiding commitment is used as hash function. This implies that the accumulated values are statistically hidden in the accumulator. A witness for an accumulated value $x$ is the opening of the corresponding commitment together with its authentication path (the authentication path only contains commitments).

*Proof.* Transformation 1 inserts a random value $x$ from the accumulation domain into the accumulator. To win the game, the adversary chooses two arbitrary sets $\mathcal{X}_0, \mathcal{X}_1$ such that $1 \leq |\mathcal{X}_0| + 1 \leq 2^t$ and $2^t < |\mathcal{X}_1| + 1$ for arbitrary integers $t > 0$. According to the game, the adversary can now obtain a membership witness for an element $x' \in \mathcal{X}_0 \cup \mathcal{X}_1$ and decide which set was accumulated by using the length $l$ of the authentication path of the witness, i.e., if $l > t$ the set $\mathcal{X}_1$ was accumulated and $\mathcal{X}_0$ otherwise. □

Since the used commitments are statistically hiding, indistinguishability holds for sets $\mathcal{X}_0, \mathcal{X}_1$ of size $2^t \leq |\mathcal{X}_0|, |\mathcal{X}_1| < 2^{t+1}$ with $t \in \mathbb{N}$, where the hash-tree is filled up with dummy elements if the set size is not a power of two. Thus, for sets of size $|\mathcal{X}_0| = |\mathcal{X}_1| = 2^t$ we have indistinguishability, whereas $\mathsf{cfw}$-indistinguishability holds for arbitrary sets meeting the aforementioned constraints.

### B.3 Indistinguishability of Universal Dynamic Accumulators

Subsequently, we prove the following lemmas by providing adversaries that succeed with non-negligible probability better than 1/2.

**Lemma 8.** *Under Transformation 1, the universal accumulator from [26] is distinguishable.*

*Proof.* Here, a non-membership witness for a value $y_j$ not contained in $\mathsf{acc}_{\mathcal{X}}$ with $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a tuple $(a, d)$, where $a \prod_{i=1}^{n} x_i + by_j = 1$ and $d \equiv g^{-b} \pmod{N}$. From this, an adversary can compute $a^{-1} \bmod y_j$, i.e., $a^{-1} \equiv \prod_{i=1}^{n} x_i \pmod{y_j}$. When $y_j > \prod_{i=1}^{n} x_i$ holds, then obviously also $a^{-1} \bmod y_j = \prod_{i=1}^{n} x_i$.

Transformation 1 inserts a random value $x$ from the accumulation domain into the accumulator. Suppose that the adversary chooses two arbitrary sets $\mathcal{X}_0 = \{x_1, \ldots, x_n\}$ and $\mathcal{X}_1 = \{w_1, \ldots, w_m\}$ with $\mathcal{X}_0 \neq \mathcal{X}_1$ and generates $k$ distinct non-membership witnesses $(y_{j_i})_{i=1}^{k} = ((a_{j_i}, d_{j_i}))_{i=1}^{k}$. The adversary can then compute $a_{j_i}^{-1} \bmod y_{j_i}$ for $i = 1, \ldots, k$ such that, depending on whether $\mathcal{X}_0$ or $\mathcal{X}_1$ was accumulated, either $x \prod_{l=1}^{n} x_l \equiv a_{j_i}^{-1} \pmod{y_{j_i}}$ or $x \prod_{l=1}^{m} w_l \equiv a_{j_i}^{-1} \pmod{y_{j_i}}$ holds for $i = 1, \ldots, k$. In the attack, the adversary can compute $a_{\mathsf{CRT}}^{-1} \pmod{\prod_{i=1}^{k} y_{j_i}}$ using $(a_{j_i}^{-1} \bmod y_{j_i})_{i=1}^{k}$ and the Chinese remainder theorem. Now, if $\prod_{i=1}^{k} y_{j_i} > x \prod_{i=1}^{n} x_i$ and $\prod_{i=1}^{k} y_{j_i} > x \prod_{i=1}^{m} w_i$, it either holds that $a_{\mathsf{CRT}}^{-1} = x \prod_{i=1}^{n} x_i$ or $a_{\mathsf{CRT}}^{-1} = x \prod_{i=1}^{m} w_i$. These conditions can always be ensured, since the adversary can generate an arbitrary number of non-membership witnesses $y_{j_i}$. Next, the adversary can divide $a_{\mathsf{CRT}}^{-1}$ by an element being exclusively contained in $\mathcal{X}_0$. If this division leaves a remainder, $\mathcal{X}_1$ was accumulated, and $\mathcal{X}_0$ otherwise. This attack succeeds with overwhelming probability, since only primes are accumulated and the probability that the randomly chosen $x$ is also exclusively contained in one of the sets $\mathcal{X}_0$ or $\mathcal{X}_1$ is negligible. □

**Lemma 9.** *Under Transformation 1, the universal accumulator from [17] is distinguishable.*

*Proof.* Again, we show that the proposed non-membership witnesses leak information about the accumulated set. Here, non-membership witnesses are of the form $(a, d)$ with $a = g^{\frac{h(s)-d}{y_j+s}} = g^{\frac{\prod_{i=1}^{n}(x_i+s)-d}{y_j+s}}$ and $d = h(-y_j) = \prod_{i=1}^{n}(x_i - y_j)$.

Transformation 1 inserts a random value $x$ from the accumulation domain into the accumulator. Suppose w.l.o.g. that the adversary chooses $\mathcal{X}_0 = \{x_1, \ldots, x_n\}$ and $\mathcal{X}_1 = \{w_1, \ldots, w_m\}$ for $m > n$ and obtains $\mathsf{acc}_{\mathcal{X}_b} \leftarrow g^{h(s)}$, where $h(s)$ is either equal to $(x+s)\prod_{i=1}^{n}(x_i+s)$ or equal to $(x+s)\prod_{i=1}^{m}(w_i+s)$. The adversary is allowed to generate $n+2$ distinct non-membership witnesses $((a_i, d_i))_{i=1}^{n+2}$ corresponding to the non-members $(y_i)_{i=1}^{n+2}$. Now, since $d_i = h(-y_i)$, one can simply recover a suspected polynomial of degree $n+1$ by its evaluations $(d_i)_{i=1}^{n+2}$ at $(-y_i)_{i=1}^{n+2}$ using polynomial interpolation. This interpolation yields the polynomial corresponding to the accumulated set in case

$\mathcal{X}_0$ was accumulated, and some arbitrary polynomial with the evaluations $(d_i)_{i=1}^{n+2}$ at $(-y_i)_{i=1}^{n+2}$ in case $\mathcal{X}_1$ was accumulated. Given this polynomial and $(g, g^s, \ldots, g^{s^t})$, one can reevaluate the accumulator, i.e., $\prod_{i=0}^{n+1}(g^{s^i})^{a_i}$, where $a_i$ is the $i$-th coefficient of the expanded polynomial. If $\mathsf{acc}_{\mathcal{X}_b} = \prod_{i=0}^{n+1}(g^{s^i})^{a_i}$ holds, then we know with overwhelming probability that $\mathsf{acc}_{\mathcal{X}_b}$ accumulates $\mathcal{X}_0$ and $\mathcal{X}_1$ otherwise. Note that this attack succeeds with overwhelming probability, since the probability that the wrong polynomial has the same evaluation at a random unknown $s$ is negligible. □

We further note that, due to the structure of the values $d$ in the non-membership witnesses, also other attacks could apply.

**Lemma 10.** *Under Transformation 1, the universal accumulators from [3] is distinguishable.*

*Proof.* We show that the proposed non-membership witnesses leak information about the accumulated set. Here, non-membership witnesses are of the form $(a, d)$ such that $a = g^{\frac{\prod_{i=1}^{n}(x_i+s)-d}{y_j+s}}$ and $d$ is the remainder of the polynomial division $\frac{\prod_{i=1}^{n}(x_i+s)}{y_j+s}$.

Transformation 1 inserts a random value $x$ from the accumulation domain into the accumulator. Similar to the attack against [26], we can w.l.o.g. assume that the adversary chooses an arbitrary set $\mathcal{X}$ and two arbitrary distinct elements $x_1, x_2 \in \mathbb{Z}_p$ such that $x_1, x_2 \notin \mathcal{X}$. Then, we have $\mathcal{X}_0 = \mathcal{X} \cup \{x_1\}$ and $\mathcal{X}_1 = \mathcal{X} \cup \{x_2\}$, respectively. According to the game, all values of $\mathcal{X}$ can be deleted. Then, the value of the challenge accumulator is $g^{(x+s)(x_i+s)}$ for an $i \in \{1, 2\}$. Furthermore, the polynomial division $\frac{(x+s)(x_i+s)}{y_j+s}$ yields a remainder of the form $xx_i - (x + x_i - y_j)y_j = d \mod p$. Now, the adversary obtains a non-membership witness for some value $y_j$, yielding the remainder $d_j$. Given that, the adversary can compute two candidate values $x', x''$ for $x$ from the equation above – simply by trying both $x_1$ and $x_2$. Thus, the adversary knows two potentially accumulated polynomials $(x' + s)(x_1 + s)$ and $(x'' + s)(x_2 + s)$, which can be used to reevaluate the accumulator with respect to these two polynomials in the same way as in the proof for Lemma 9. Thus, the adversary can decide which set has been accumulated by comparing the resulting accumulators to the challenge accumulator. This attack succeeds with overwhelming probability, since the probability that the wrong polynomial has the same evaluation at $s$ is negligible for a random, unknown $s$. □

We note that in [3] a second method for generating non-membership witnesses (where the knowledge of $s$ is required) is proposed. Here, $d$ is reduced modulo $y_j + s$, which, in turn, means that the attack above only succeeds when $d \bmod p = d \bmod y_j + s$. However, since $d$ leaks information about the set, also other, more efficient attacks might be possible.

**Lemma 11.** *Under Transformation 1, the universal accumulators from vector commitments [15] are distinguishable.*

*Proof.* We show that the proposed accumulator schemes from vector commitments in the RSA and CDH setting are distinguishable by presenting an adversary that wins the indistinguishability game with a probability of 1.

In accumulators from vector commitments, the accumulation domain is the set $D = \{1, \ldots, t\}$. In both, the RSA and the CDH instantiation, the accumulator (vector

commitment) for a set $\mathcal{X} \subseteq D$ is computed as $\mathsf{acc}_{\mathcal{X}} \leftarrow \prod_{i=1}^{t} g_i^{\chi_{\mathcal{X}}(i)}$, where $\chi_{\mathcal{X}}(\cdot)$ is the characteristic function. The values $g_i$ are contained in the public parameters.

The adversary can choose two arbitrary sets $\mathcal{X}_0, \mathcal{X}_1$, such that $|\mathcal{X}_0 \cap \mathcal{X}_1| \leq |\mathcal{X}_0| - 2$ and $|\mathcal{X}_0 \cap \mathcal{X}_1| \leq |\mathcal{X}_1| - 2$, i.e., $\mathcal{X}_0$ and $\mathcal{X}_1$ are different in at least two elements. Transformation 1 inserts an additional random value from the accumulation domain into the accumulator. Then, the value of $\mathsf{acc}_{\mathcal{X}_b}$ is either equal to $g_r \prod_{i \in \mathcal{X}_0} g_i$ with $r \notin \mathcal{X}_0$, or equal to $g_{r'} \prod_{i \in \mathcal{X}_1} g_i$ with $r' \notin \mathcal{X}_1$. Note that due to the choice of the sets $\mathcal{X}_0$ and $\mathcal{X}_1$ we can guarantee that adding a single element cannot make the sets collide. Consequently, the adversary can w.l.o.g. try if an element from the set $D \setminus \mathcal{X}_0$ was used as randomizer for evaluating $\mathsf{acc}_{\mathcal{X}_b}$ with respect to $\mathcal{X}_0$. If such an element is found, $\mathcal{X}_0$ was accumulated and $\mathcal{X}_1$ otherwise. Note that the size of the public parameters is linear in the size of $D$. Thus, this brute-force strategy can be realized efficiently. □

We note that the same argumentation can be applied if the accumulation domain is an arbitrary set of size $n$ (as proposed in Remark 16 in [15]).

**Lemma 12.** *Under Transformation 1, the universal accumulators in [8], [9] and [11] are distinguishable.*

*Proof.* We show that the accumulators from collision-resistant hashing are distinguishable by presenting an adversary that wins the indistinguishability game with probability 1. Here, one can use the fact that the sets are sorted and that non-membership witnesses contain two elements of the respective sets. Again, Transformation 1 inserts an additional random element into the set upon $\mathsf{Eval}$.

To break indistinguishability, the adversary chooses two disjoint sets $\mathcal{X}_0, \mathcal{X}_1$. It is allowed to obtain a non-membership witness for a value $y_i \notin \mathcal{X}_0 \cup \mathcal{X}_1$, which contains two consecutive values $x_1, x_2$ out of the accumulated set such that $x_1 < y_i < x_2$ holds.

Then we distinguish three cases. In the first case $x_1$ and $x_2$ are contained in $\mathcal{X}_i$ for an $i \in \{0, 1\}$, meaning that the adversary knows that $\mathcal{X}_i$ has been accumulated (neither $x_1$ nor $x_2$ is the random element). In the second case, we can w.l.o.g. assume that $x_2 \notin \mathcal{X}_0 \cup \mathcal{X}_1$ (is the random element which is contained in none of the initially chosen sets). This means that $x_1 \in \mathcal{X}_i$ for an $i \in \{0, 1\}$ and the adversary knows that $\mathcal{X}_i$ was accumulated. In the remaining case, we can w.l.o.g. assume that $x_1 \in \mathcal{X}_0$ and $x_2 \in \mathcal{X}_1$, and, thus, either $x_1$ is the random element added to $\mathcal{X}_1$ upon $\mathsf{Eval}$ or vice versa. Then, the adversary can request another non-membership witness for $y_i' < x_1$ and $y_i' \notin \mathcal{X}_0 \cup \mathcal{X}_1$ and obtains $x_3$ and $x_4$ such that $x_3 < y_i' < x_4$. It could be the case that $x_1 = x_4$, but still $x_3$ is either an element of $\mathcal{X}_0$ or $\mathcal{X}_1$ and, thus, the adversary can decide which set was accumulated.

Finally, we note that the adversary can always choose $y_i$ and $y_i'$ such that obtaining witnesses fulfilling the requirements above is possible, since $\mathcal{X}_0$ and $\mathcal{X}_1$ are also chosen by the adversary. □

# C Proofs

## C.1 Undeniability Implies Collision-Freeness

We prove that every undeniable accumulator is also collision-free by showing that an efficient adversary $\mathcal{A}$ against the *collision freeness* of an accumulator scheme can be used to construct an efficient adversary $\mathcal{B}$ against its *undeniability*. Note that $\mathcal{B}$ can simulate all required oracles.

*Proof.* $\mathcal{B}$ gets input $\mathsf{pk}_{\mathsf{acc}}$ and runs $\mathcal{A}(\mathsf{pk}_{\mathsf{acc}})$. There are two cases if $\mathcal{A}$ wins the collision freeness game: $\mathcal{A}$ outputs either $(\mathsf{wit}^*_{x_i}, x_i^*, \mathcal{X}^*, r^*)$ or $(\underline{\mathsf{wit}}^*_{x_i}, x_i^*, \mathcal{X}^*, r^*)$.

If $\mathcal{A}$ outputs $(\mathsf{wit}^*_{x_i}, x_i^*, \mathcal{X}^*, r^*)$ such that $x_i^* \notin \mathcal{X}^*$, $\mathcal{B}$ evaluates the accumulator with respect to $\mathcal{X}^*$ and $r^*$. Hence, it obtains $\mathsf{acc}_{\mathcal{X}^*}$. Next, it computes $\underline{\mathsf{wit}}_{x_i}$ using the witness generation algorithm (which can always be done since $x_i \notin \mathcal{X}$) and outputs $(\mathsf{wit}^*_{x_i}, \underline{\mathsf{wit}}_{x_i}, x_i^*, \mathsf{acc}_{\mathcal{X}^*})$.

If $\mathcal{A}$ outputs $(\underline{\mathsf{wit}}^*_{x_i}, x_i^*, \mathcal{X}^*, r^*)$ such that $x_i^* \in \mathcal{X}^*$, $\mathcal{B}$ evaluates the accumulator with respect to $\mathcal{X}^*$ and $r^*$. Hence, it obtains $\mathsf{acc}_{\mathcal{X}^*}$. Next, it computes $\mathsf{wit}_{x_i}$ using the witness generation algorithm (which can always be done since $x_i^* \in \mathcal{X}^*$) and outputs $(\mathsf{wit}_{x_i}, \underline{\mathsf{wit}}^*_{x_i}, x_i^*, \mathsf{acc}_{\mathcal{X}^*})$. Hence, whenever $\mathcal{A}$ wins the collision freeness game, $\mathcal{B}$ wins the undeniability game with exactly the same probability □

### C.2 Commitments from Indistinguishable 1-bounded Accumulators

*Proof.* We show that an efficient adversary $\mathcal{A}$ against the binding (hiding) property of Scheme 1 can be turned into an adversary $\mathcal{B}$ against the collision freeness (indistinguishability) of the indistinguishable 1-bounded accumulator. To do so, we construct a reduction $\mathcal{B}$, which interacts with a challenger $\mathcal{C}$ from the respective accumulator game (and internally simulates the challenger $\mathcal{C}'$ of the respective commitment game for adversary $\mathcal{A}$).

*(Binding):* Let us assume that there exists an efficient adversary $\mathcal{A}$ against the binding property of Scheme 1. $\mathcal{C}$ runs $\mathsf{pk}_{\mathsf{acc}} \leftarrow \mathsf{Acc.Gen}(1^\kappa, 1)$ and starts $\mathcal{B}$ on $\mathsf{pk}_{\mathsf{acc}}$. That is, $\mathcal{C}'$ sets $\mathsf{pp} \leftarrow \mathsf{pk}_{\mathsf{acc}}$ and starts $\mathcal{A}$ on input $\mathsf{pp}$. Eventually, $\mathcal{A}$ outputs $C^*, O^* \leftarrow (r, m, \mathsf{wit}_m, \mathsf{aux}), O'^* \leftarrow (r', m', \mathsf{wit}_{m'}, \mathsf{aux}')$ such that $m \leftarrow \mathsf{Open}(\mathsf{pp}, C^*, O^*)$ and $m' \leftarrow \mathsf{Open}(\mathsf{pp}, C^*, O'^*)$ and $m \neq m' \ \wedge \ m \neq \bot \ \wedge \ m' \neq \bot$. Note that this means that $\mathsf{Eval}_r((\emptyset, \mathsf{pk}_{\mathsf{acc}}), \{m\}) = C^*$, $\mathsf{Eval}_{r'}((\emptyset, \mathsf{pk}_{\mathsf{acc}}), \{m'\}) = C^*$, $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, C^*, \mathsf{wit}_m, m) = \mathsf{true}$ and $\mathsf{Verify}(\mathsf{pk}_{\mathsf{acc}}, C^*, \mathsf{wit}_{m'}, m') = \mathsf{true}$. Thus, $\mathcal{B}$ returns $(\mathsf{wit}_m, m, \{m'\}, r')$ as a collision for the accumulator.

*(Hiding):* Let us assume that there exists an efficient adversary $\mathcal{A}$ against the hiding property of Scheme 1. $\mathcal{C}$ runs $\mathsf{pk}_{\mathsf{acc}} \leftarrow \mathsf{Acc.Gen}(1^\kappa, 1)$ and starts $\mathcal{B}$ on $\mathsf{pk}_{\mathsf{acc}}$, meaning that $\mathcal{C}'$ sets $\mathsf{pp} \leftarrow \mathsf{pk}_{\mathsf{acc}}$, runs $(m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{pp})$ and returns $(\mathcal{X}_0, \mathcal{X}_1, \mathsf{state}) \leftarrow (\{m_0\}, \{m_1\}, \mathsf{state})$. $\mathcal{C}$ then computes the challenge accumulator $\mathsf{acc}_{\mathcal{X}_b}$ and hands it to $\mathcal{B}$. Given that, $\mathcal{C}'$ starts $\mathcal{A}(\mathsf{pp}, \mathsf{acc}_{\mathcal{X}_b}, \mathsf{state})$ and obtains and outputs $b^*$. Thus, $\mathcal{B}$ breaks the indistinguishability of the accumulator. □

### C.3 Indistinguishable Undeniable Accumulators from Zero-Knowledge Sets

*Proof.* It is easy to see that the notions perfect completeness and soundness are equivalent to the correctness and undeniability notions of accumulators. However, the zero-knowledge property of ZK-sets is defined in a simulation-based way, whereas the indistinguishability is defined in a game-based way. To show that the zero-knowledge property of ZK-sets implies indistinguishability, we use the following sequence of games.

As a ZK-set is zero-knowledge, there is a simulator $\mathcal{S}$ whose output cannot be distinguished from an honest output with non-negligible probability $\epsilon(\kappa)$. We will now use $\mathcal{S}$ to replace the oracle answers in the indistinguishability game.

**Game** 0**:** The original accumulator indistinguishability game.

**Game 1:** We change the game such that all relevant steps are executed by the simulator. That is, the setup ($\mathsf{Gen}$) is performed using $\mathcal{S}^{\mathsf{G}}$ and the computation of $\mathsf{acc}_{\mathcal{X}_b}$ is performed by the simulator using $\mathcal{S}^{\mathsf{E}}$. Furthermore, the output of oracle $\mathcal{O}^{\mathsf{E}}$ is replaced by the output of $\mathcal{S}^{\mathsf{E}}$. Likewise, the outputs of the oracles $\mathcal{O}^{\mathsf{W}}$ and $\mathcal{O}^{\underline{\mathsf{W}}}$ are replaced by the output of $\mathcal{S}^{\mathsf{Q}}$. More precisely, if $\mathcal{O}^{\mathsf{W}}$ has been called and $\mathcal{S}^{\mathsf{Q}}$ returns a membership proof then $\mathcal{O}^{\mathsf{W}}$ returns $\pi$ and $\perp$ otherwise. The oracle $\mathcal{O}^{\underline{\mathsf{W}}}$ works in the same way.

Henceforth, we denote the winning condition of Game $i$ by $S_i$. In Game 1, the adversary only gets to see simulated values, which contain no information about the respective set. Therefore, the advantage for the adversary to win this game is equal to 0, i.e., $\mathsf{Pr}[S_1] = \frac{1}{2}$. We further know that the adversary can only recognize the transition from Game 0 to Game 1 with probability $\epsilon(\kappa)$. Hence, we know that $|\mathsf{Pr}[S_0] - \mathsf{Pr}[S_1]| = \epsilon(\kappa)$. Taking everything together, we derive that $|\mathsf{Pr}[S_0] - \frac{1}{2}| = \epsilon(\kappa)$ which shows that every ZK-set fulfilling the zero knowledge property is also an indistinguishable accumulator. □