Zero-Knowledge Accumulators and Set Algebra

Esha Ghosh[1](✉), Olga Ohrimenko[2], Dimitrios Papadopoulos[3],
Roberto Tamassia[1], and Nikos Triandopoulos[4]

[1] Department of Computer Science, Brown University, Providence, USA
esha_ghosh@brown.edu, rt@cs.brown.edu
[2] Microsoft Research, Cambridge, UK
oohrim@microsoft.com
[3] University of Maryland, College Park, USA
dipapado@umd.edu
[4] Stevens Institute of Technology, Hoboken, USA
ntriando@stevens.edu

Abstract. Cryptographic accumulators allow to succinctly represent a set by an accumulation value with respect to which short (non-)membership proofs about the set can be efficiently constructed and verified. Traditionally, their security captures soundness but offers no privacy: Convincing proofs reliably encode set membership, but they may well leak information about the accumulated set.

In this paper we put forward a strong privacy-preserving enhancement by introducing and devising *zero-knowledge accumulators* that additionally provide hiding guarantees: Accumulation values and proofs leak nothing about a dynamic set that evolves via element insertions/deletions. We formalize the new property using the standard real-ideal paradigm, namely demanding that an adaptive adversary with access to query/update oracles, cannot tell whether he interacts with honest protocol executions or a simulator fully ignorant of the set (even of the type of updates on it). We rigorously compare the new primitive to existing ones for privacy-preserving verification of set membership (or other relations) and derive interesting implications among related security definitions, showing that zero-knowledge accumulators offer stronger privacy than recent related works by Naor et al. [TCC 2015] and Derler et al. [CT-RSA 2015]. We construct the first dynamic universal zero-knowledge accumulator that we show to be perfect zero-knowledge and secure under the q-Strong Bilinear Diffie-Hellman assumption.

Finally, we extend our new privacy notion and our new construction to provide privacy-preserving proofs also for an authenticated dynamic set collection—a primitive for efficiently verifying more elaborate set operations, beyond set-membership. We introduce a primitive that supports a *zero-knowledge verifiable set algebra*: Succinct proofs for union, intersection and set difference queries over a dynamically evolving collection of sets can be efficiently constructed and optimally verified, while—for the first time—they leak nothing about the collection beyond the query result.

© International Association for Cryptologic Research 2016
J.H. Cheon and T. Takagi (Eds.): ASIACRYPT 2016, Part II, LNCS 10032, pp. 67–100, 2016.
DOI: 10.1007/978-3-662-53890-6_3

Keywords: Zero-knowledge dynamic and universal accumulators · Zero-knowledge updates · Zero-knowledge set algebra · Outsourced computation · Integrity · Privacy · Bilinear accumulators · Cloud privacy

1 Introduction

A cryptographic accumulator is a primitive that offers a way to succinctly represent a set of elements $\mathcal{X}$ by a single value acc referred to as the *accumulation value*. Moreover, it provides a method to efficiently and succinctly prove (to a party that only holds acc) that an element x belongs to $\mathcal{X}$, by computing a constant-size proof w, referred to as *witness*. The interaction is in a three-party model, where the trusted *owner* of the set runs the initial key generation and setup process to publish the accumulation value. Later an untrusted *server* handles queries on the set issued by *clients*, providing membership answers with publicly verifiable witnesses.

Accumulators were originally introduced by Benaloh and del Mare in [4]. Numerous constructions have been proposed since, operating in various models [2,3,7,9–12,19,23,44,54][1]. Most notably, the primitive was extended to support non-membership witnesses [2,19,41], and efficient updates [2,11], introducing the notion of *universal* and *dynamic accumulators*, respectively. At the same time, accumulators found numerous other applications in the context of public-key infrastructure, certificate management and revocation, time-stamping, authenticated dictionaries, set operations, authenticated database queries, anonymous credentials, and more.

Traditionally in the literature, the security property associated with accumulators is *soundness* (or *collision-freeness*), expressed as the inability to forge a witness for an element, i.e., if $x \notin \mathcal{X}$, it should be hard to prove $x \in \mathcal{X}$ (and vice-versa for universal accumulators). No notion of privacy was considered until recently [20,23], e.g., "does the accumulation reveal anything about the elements of $\mathcal{X}$" or "what can an adversarial client, that asks queries and is presented with the accumulation and witnesses, learn about the set $\mathcal{X}$". It is clear that such a property would be attractive, if not—depending on the application—crucial. For example, in the context of securing the Domain Name System (DNS) protocol by accumulating the set of records in a zone, it is crucial to leak no information about values in the accumulated set while responding to queries.[2] As additional examples, Miers et al. [49] developed a privacy enhancement for Bitcoin, that utilizes the accumulator from [11], while Hanser and Slamanig [35] used accumulators to build randomizable polynomial commitments for anonymous credentials. In such a context, it is very important to minimize what is leaked by accumulation values and witnesses in order to achieve anonymity (for individuals and transactions).

In this work, we propose the notion of *zero-knowledge* for cryptographic accumulators. We define this property via an extensive real/ideal game, similar to

[1] We refer interested readers to [23] for a comprehensive review of existing schemes.

[2] See for example, https://tools.ietf.org/html/rfc5155.

that of standard zero-knowledge [34]. In the real setting, an adversary is allowed to choose his challenge set and to receive the corresponding accumulation. He is then given oracle access to the querying algorithm as well as an update algorithm that allows him to request updates in the set (receiving the updated accumulation value every time). In the ideal setting, the adversary interacts with a simulator that does not know anything about the set or the nature of the updates, other than the fact that an update occurred. Zero-knowledge is then defined as the inability of the adversary to distinguish between the two settings.

We provide the first zero-knowledge accumulator construction and prove its security. Our construction builds upon the *bilinear accumulator* of Nguyen [53] and achieves *perfect* zero-knowledge. Our scheme falls within the category of *dynamic universal* cryptographic accumulators: It allows to prove both membership and non-membership statements (i.e., one can compute a witness for $x \notin \mathcal{X}$), and supports efficient updates in the accumulation value due to insertions and deletions in the set. It satisfies soundness under the q-Strong Bilinear Diffie-Hellman assumption (q-SBDH), introduced in [6]. In order to provide non-membership witness computation in zero-knowledge, we had to deviate from existing non-membership proof techniques for the bilinear accumulator [2,19]. We instead use the set-disjointness technique of [56], appropriately enhanced for privacy. From an efficiency perspective, we show that introducing zero-knowledge to the bilinear accumulator comes at an insignificant cost: Asymptotically all query overheads are either the same or within a poly-logarithmic factor of the construction in [53] that offers no privacy.

Zero-knowledge vs. indistinguishability. Recently, de Meer et al. [20] and Derler et al. [23] introduced an *indistinguishability* property for cryptographic accumulators. Unfortunately, the definition of the former was inherently flawed, as noted in [23].[3] The accumulator definition in [23], while meant to support changes in the accumulated set (i.e., element insertion or deletion), did not protect the privacy of these changes. In particular, any adversary suspecting a particular modification in the set could easily check the correctness of his guess. Our notion of zero-knowledge differs from the privacy notion of [23], by protecting not only the originally accumulated set but also all subsequent updates. In fact, we formally prove that, for cryptographic accumulators, zero-knowledge is a strictly stronger property than indistinguishability.

Relation to zero-knowledge sets. Our privacy notion is reminiscent of that of zero-knowledge sets [14,15,43,48,58] where set membership and non-membership queries can be answered without revealing anything else about the set. Zero-knowledge accumulators can be seen as a relaxation of zero-knowledge sets in an "honest-committer" setting. In Sect. 3.2 we discuss this relation in more detail, also looking into the dynamic setting, comparing with existing work on updatable zero-knowledge sets [45].

[3] Subsequently, the definition was strengthened in [61], but it is still subsumed by that of [23].

Relation to zero-knowledge authenticated data structures. A cryptographic accumulator can be viewed as a special case of an *authenticated data structure* (ADS) [51,63], where the supported data type is a set. Likewise, the zero-knowledge accumulator we introduce here, falls within the framework of *zero-knowledge authenticated data structures* (ZKADS) introduced recently in [29]. We discuss the relation in detail in Sect. 3.2.

Beyond set-membership. One question that arises naturally is how to build a "set-friendly" ZKADS with a supported functionality beyond set-membership. In particular, given multiple sets, we are interested in accommodating more elaborate set-operations: set union, intersection and difference.[4] We introduce the primitive of *zero-knowledge authenticated dynamic set collection* for the following setting. A party that owns a database of sets outsources it to an untrusted server that is subsequently charged with handling queries, expressed as set operations among the database sets, issued by multiple clients; at any point, the owner can make updates to the outsourced sets. We present the first scheme that provides not only integrity of computations but also privacy for the queried set (i.e., the provided proofs leak nothing beyond the answer). The basic building block is our zero-knowledge accumulator, together with a carefully deployed *accumulation tree* [57]. We note that if we restrict the security properties only to soundness—as is the case in the traditional literature of ADS—there are existing schemes (specifically for set-operations) by Papamanthou et al. [56] for the single-operation case, and by Canetti et al. [12] and Kosba et al. [40] for the case of multiple (nested) operations. However, none of these constructions offers privacy, thus our scheme is a natural strengthening of their security guarantees, while maintaining the same efficient performance. Preserving efficiency while maintaining integrity and zero-knowledge privacy turned out to be quite challenging. In particular, answering union and set difference queries for set collections required new techniques to be developed. At a high level, the efficiency of the proof techniques in [56] strongly relies on revealing much of the non-queried information and hence could not be extended to support privacy-preserving queries.

Contributions. Our contributions can be summarized as follows:

- We introduce the property of zero-knowledge for cryptographic accumulators and show that it is strictly stronger than existing privacy notions for accumulators.
- We give an overview of the connection between zero-knowledge accumulators and related cryptographic primitives in the area (e.g., we show that zero-knowledge accumulators imply primary-secondary-resolver systems proposed in [52]).
- We provide the first construction of a zero-knowledge dynamic universal accumulator. Our scheme is perfect zero-knowledge and secure under the q-SBDH assumption; it achieves these security properties with only a small

[4] We stress that, in the computational setting, these operations form a complete set algebra.

(or no) overhead. We compare efficiency with the accumulator of [53] in Fig. 3 in terms of number of cryptographic operations performed.

- Using our zero-knowledge accumulator as a building block, we construct the first protocol for zero-knowledge outsourced set operations. Our scheme is non-interactive and offers secure and efficient subset, intersection and union operations under the q-SBDH assumption. For set-difference queries, our construction is secure under q-SBDH assumption as well, but proof construction entails a Sigma protocol thus requiring interaction. This secure set-difference protocol can also be made non-interactive, albeit in the random oracle model, (in which case the construction is in the Common Reference String model). Our construction (except for the update cost) is asymptotically as efficient as the previous state-of-the-art construction from [56], that offered no privacy guarantees.

1.1 Other Related Work

Existing works (e.g., [2,11,41,53]) equip some accumulators with zero-knowledge proof-of-knowledge protocols, such that a party that knows that value x is (or is not) in $\mathcal{X}$ can efficiently prove it to a third-party arbitrator, without revealing the value. While hiding x, all existing constructions trivially expose the accumulation value as part of the proven statement. This may itself reveal information about set $\mathcal{X}$. Our privacy goals are therefore different, yet the techniques are compatible. Developing zero-knowledge proof-of-knowledge protocols for membership and non-membership, that can work with a zero-knowledge accumulator will yield a strong tool that leaks nothing about either the set or the particular element in the proof.

Most widely-used accumulator constructions—including ours—are in the trusted-setup model, i.e., the party that generates the scheme parameters originally, holds some trapdoor information that is not revealed to the adversary. E.g., for the RSA-based constructions, any adversary that knows the factorization of the modulo can trivially cheat. An alternative body of work aims to build *trapdoorless* accumulators (also referred to as *strong* accumulators) [7,9,44,54,55,62], where the owner is entirely untrusted (effectively the owner and the server are the same entity). Unfortunately, the earlier of these works are quite inefficient for all practical purposes, while the more recent ones either yield witnesses that grow logarithmically with the size of $\mathcal{X}$ or rely on algebraic groups that are not yet well-studied in cryptography. A straight-forward way to construct a strong accumulator is via a black-box reduction from zero-knowledge sets (with corresponding efficiency caveats). While a scheme without the need for a trusted setup is clearly more attractive in terms of security, it is safe to say that we do not yet have a practical scheme with constant-size proofs, based on standard security assumptions.

Recently, Naor et al. [52] introduced primary-secondary-resolver membership proof systems, a primitive that is also a relaxation of zero-knowledge sets in the three-party model, and showed applications in network protocols [33]. While our definitions have similarities, in Sect. 3.2 we show that zero-knowledge accumulators are a stronger primitive than primary-secondary-resolver systems.

Regarding related work for set operations, the focus in the cryptographic literature has been on the privacy aspect with a very long line of works (see for example, [5,27,36,37,39]), some of which focus specifically on set-intersection (e.g., [17,18,24,38]). The above works fit in the secure two-party computation model and most are secure (or can be made with some loss in efficiency) also against malicious adversaries, thus guaranteeing the authenticity of the result. However, this approach typically requires multi-round interaction or larger communication cost than our construction. On the other hand, our two security properties are "one-sided": Only the server may cheat with respect to soundness and only the client with respect to privacy; in this setting we achieve non-interactive solutions with optimal proof-size. There also exist works that deal exclusively with the integrity of set operations, such as [50] that achieves linear verification and proof cost, and [64] that only focuses on set-intersection but can be combined with an encryption scheme to achieve privacy versus the server.

Another work that is related to ours is that of Fauzi et al. [25] that presents an efficient non-interactive zero-knowledge argument for proving relations between committed sets. Conceptually this work is close to zero-knowledge sets, allowing also for more general set operation queries. From a security viewpoint, it is in the stronger two-party model and, from a functionality viewpoint, it works for (more general) multi-set operations. However, its security relies on non-falsifiable knowledge assumptions, and the construction trivially leaks an upper-bound on the committed sets. Moreover, it cannot be efficiently generalized for operations on more than two sets at a time and it does not explicitly support efficient modifications in the sets.

We also note that recently other instantiations of zero-knowledge authenticated data structures have been proposed, including lists, trees and partially-ordered sets of bounded dimension [29,32].

2 Preliminaries

We denote with λ the security parameter and with $\nu(\lambda)$ a negligible function. A function $f(\lambda)$ is negligible if for each polynomial function $poly(\lambda)$ and all large enough values of λ, $f(\lambda) < 1/(poly(\lambda))$. We say that an event can occur with negligible probability if its occurrence probability can be upper bounded by a negligible function. Respectively, an event takes place with overwhelming probability if its complement takes place with negligible probability. The symbol $\overset{\$}{\leftarrow} \mathbb{X}$ denotes uniform sampling from domain $\mathbb{X}$. We denote the fact that a party Adv (instantiated as Turing machine) is probabilistic and runs in polynomial-time by writing PPT Adv.

Bilinear pairings. Let $\mathbb{G}$ be a cyclic multiplicative group of prime order p, generated by g. Let also $\mathbb{G}_T$ be a cyclic multiplicative group with the same order p and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear pairing with the following properties:

(1) Bilinearity: $e(P^a, Q^b) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$; (2) Non-degeneracy: $e(g, g) \neq 1_{\mathbb{G}_T}$; (3) Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}$. We denote with $pub := (p, \mathbb{G}, \mathbb{G}_T, e, g)$ the bilinear pairings parameters, output by a randomized polynomial-time algorithm $\mathsf{GenParams}$ on input 1^λ. For clarity of presentation, we assume for the rest of the paper a symmetric (Type 1) pairing e. We note though that both our constructions can be securely implemented in the (more efficient) asymmetric pairing case, with straight-forward modifications (see [16] for a general discussion on pairings). Our security proofs make use of the q-Strong Bilinear Diffie-Hellman (q-SBDH) assumption over groups with bilinear pairings introduced in [6].

Assumption 1 (q-Strong Bilinear Diffie-Hellman). *For any PPT adversary* Adv *and for q being a parameter of size polynomial in λ, there exists negligible function $\nu(\lambda)$ such that the following holds:*

$$\Pr\left[\begin{array}{c} pub \leftarrow \mathsf{GenParams}(1^\lambda); s \leftarrow_R \mathbb{Z}_p^*; \\ (z, \gamma) \in \mathbb{Z}_p^* \times \mathbb{G}_T \leftarrow \mathsf{Adv}(pub, (g^s, ..., g^{s^q})) : \gamma = e(g, g)^{1/(z+s)} \end{array}\right] \leq \nu(\lambda) .$$

Complexity model. For ease of notation, we measure the asymptotic performance of our schemes by counting numbers of operations and group elements, ignoring a, poly-logarithmic in λ, factor (e.g., an operation in $\mathbb{G}$ takes one unit time).

Characteristic polynomial. A set $\mathcal{X} = \{x_1, \ldots, x_n\}$ with elements $x_i \in \mathbb{Z}_p$ can be represented by a polynomial following an idea introduced in [27]. The polynomial $\mathsf{Ch}_{\mathcal{X}}(z) = \prod_{i=1}^n (x_i + z)$ from $\mathbb{Z}_p[z]$, where z is a formal variable, is called the *characteristic polynomial* of $\mathcal{X}$. In what follows, we will denote this polynomial simply by $\mathsf{Ch}_{\mathcal{X}}$ and its evaluation at a point y as $\mathsf{Ch}_{\mathcal{X}}(y)$. Characteristic polynomials enjoy a number of homomorphic properties w.r.t. set operations. We use the following characterization of set intersection of the sets: Given a collection of sets $\mathcal{X}_{i_1}, \ldots \mathcal{X}_{i_k}$ and their characteristic polynomial representation, we summarize a characterization of the intersection of the sets in the following lemma.

Lemma 1 ([56]). *A set* answer *that is a common subset of sets $\mathcal{X}_{i_1}, \ldots \mathcal{X}_{i_k}$, is their intersection if and only if there exist polynomials $q_1[z], \ldots q_k[z]$ such that $\sum_{j \in [i_1, i_k]} q_j[z] P_j[z] = 1$ where $P_j[z] = \mathsf{Ch}_{\mathcal{X}_j \setminus \mathsf{answer}}[z]$. Computing polynomials $q_j[z]$ where $j \in [i_1, i_k]$ has $O(N \log^2 N \log\log N)$ complexity where $N = \sum_{j \in [i_1, i_k]} n_j$ and $n_j = |\mathcal{X}_j|$.*

The following two lemmas characterize the efficiency of computing the characteristic polynomial of a set –note that there is no requirement for the existence of an n-th root of unity in $\mathbb{Z}_p$ for such an algorithm to exist– and the probability that two polynomials are equivalent at a randomly chosen point.

Lemma 2 ([59]). *Given a set $\mathcal{X} = x_1, ..., x_n \in \mathbb{Z}_p^n$, its characteristic polynomial* $\mathsf{Ch}_{\mathcal{X}} := \sum_{i=0}^n c_i z^i \in \mathbb{Z}_p[z]$ *can be computed with $O(n \log n)$ operations by FFT interpolation.*

Lemma 3 (Schwartz–Zippel–DeMillo–Lipton). *Let $p[z], q[z]$ be two d-degree polynomials from $\mathbb{Z}_p[z]$ with $p[z] \neq q[z]$, Then for $w \xleftarrow{\$} \mathbb{Z}_p$, the probability that $p(w) = q(w)$ is at most d/p, and the equality can be tested in time $O(d)$.*

If $p \in O(2^\lambda)$, it follows that the above probability is negligible, if d is $poly(\lambda)$.

Accumulation tree. Given a collection of sets $\mathbb{S} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m\}$, let $\mathsf{acc}(\mathcal{X}_i)$ be a succinct representation of $\mathcal{X}_i$ using its characteristic polynomial. We describe an authentication mechanism that does the following. A trusted party computes m hash values $h_i := h(\mathsf{acc}(\mathcal{X}_i))$ (using a collision resistant cryptographic hash function) of the m sets of $\mathbb{S}$. Then given a short public digest information of the current set collection $\mathbb{S}$, the authentication mechanism provides publicly verifiable proofs of the form "h_i is the hash of the i^{th} set of the current set collection $\mathbb{S}$". A popular authentication mechanism for such proofs are Merkle hash trees [47] based on a single value digest can provide logarithmic size proofs and support updates. An alternative mechanism to Merkle trees, (specifically in the bilinear group setting) are *accumulation trees* [57]. Intuitively, an accumulation tree can be seen as a "flat" version of Merkle trees. In this work, we use our extension (for batch updates) of the accumulation tree in [56]. The detailed construction can be found in the full version.

3 Zero-Knowledge Universal Accumulators (ZKUA)

A *dynamic universal accumulator* (DUA) consists of five probabilistic polynomial time algorithms (GenKey, Setup, $\mathsf{Witness}$, Verify, Update). It represents a set $\mathcal{X}$, with elements from domain $\mathbb{X}$, by an accumulation value $\mathsf{acc} \in \mathbb{A}$. It supports queries of the form "is $x \in \mathcal{X}$?" for $x \in \mathbb{X}$ and updates to the current set (e.g., using "insert x" or "remove x" operations). The algorithms of DUA, as described below, are run between three parties: the owner, the server and the client. We follow the definitional style of [23,26] where the accumulator is described as a tuple of algorithms. In the full version we provide a discussion regarding our chosen definitional style.

Definition 1 ***(Dynamic Universal Accumulator).*** *A dynamic universal accumulator is a tuple of five PPT algorithms,* $\mathsf{DUA} = (\mathsf{GenKey}, \mathsf{Setup}, \mathsf{Witness}, \mathsf{Verify}, \mathsf{Update})$ *defined as follows:*

$(sk, vk) \leftarrow \mathsf{GenKey}(1^\lambda)$

This probabilistic algorithm takes as input the security parameter and outputs a (public) verification key vk that will be used by the client to verify query responses and a secret key sk that is kept by the owner.

$(\mathsf{acc}, ek, \mathsf{aux}) \leftarrow \mathsf{Setup}(sk, \mathcal{X})$

This probabilistic algorithm is run by the owner. It takes as input the source set $\mathcal{X}$ and produces the accumulation value acc *that will be published to both server and client, and an evaluation key ek as well as auxiliary information* aux *that will be sent only to the server in order to facilitate proof construction.*

$(b, \mathsf{w}) \leftarrow \mathsf{Witness}(\mathsf{acc}, \mathcal{X}, x, ek, \mathsf{aux})$
This algorithm is run by the server. It takes as input the evaluation key and the accumulation value ek, acc *generated by the owner, the source set* $\mathcal{X}$*, a queried element* x*, as input. It outputs a boolean value* b *indicating whether the element is in the set and a witness* w *for the answer.*

$(\mathsf{accept/reject}) \leftarrow \mathsf{Verify}(\mathsf{acc}, x, b, \mathsf{w}, vk)$
This algorithm is run by the client. It takes as input the accumulation value acc *and the public key* vk *computed by the owner, a queried element* x*, a bit* b*, the witness* w *and it outputs* $\mathsf{accept/reject}$*.*

$(\mathsf{acc}', ek', \mathsf{aux}') \leftarrow \mathsf{Update}(\mathsf{acc}, \mathcal{X}, x, sk, \mathsf{aux}, \mathsf{upd})$
This algorithm takes as input the current set with its accumulation value and auxiliary information, as well as an element x *to be inserted to* $\mathcal{X}$ *if* $\mathsf{upd} = 1$ *or removed from* $\mathcal{X}$ *if* $\mathsf{upd} = 0$*. If* $\mathsf{upd} = 1$ *and* $x \in \mathcal{X}$*, (likewise if* $\mathsf{upd} = 0$ *and* $x \notin \mathcal{X}$*) the algorithm outputs* $\perp$ *and halts, indicating an invalid update. Otherwise, it outputs* $(\mathsf{acc}', ek', aux')$ *where* acc' *is the new accumulation value corresponding to set* $\mathcal{X} \cup \{x\}$ *or* $\mathcal{X} \setminus \{x\}$ *(to be published),* ek' *is the (possibly) modified evaluation key, and* aux' *is respective auxiliary information (both to be sent only to the server).*

To update existing witnesses efficiently (i.e., not recomputing them from scratch) after a change of the accumulation value, we define the $\mathsf{WitUpdate}$ functionality.

$(\mathsf{upd}, \mathsf{w}') \leftarrow \mathsf{WitUpdate}(\mathsf{acc}, \mathsf{acc}', x, \mathsf{w}, y, ek', \mathsf{aux}, \mathsf{aux}', \mathsf{upd})$

This algorithm is to be run after an invocation of Update*. It takes as input the old and the new accumulation values and auxiliary informations, the evaluation key* ek' *output by* Update *, as well as the element* x *that was inserted or removed from the set, according to the binary value* upd *(the same as in the execution of* Update *). It also takes a different element* y *and its existing witness* w *(that may be a membership or non-membership witness). It outputs a new witness* w' *for* y*, with respect to the new set* $\mathcal{X}'$ *. The output must be the same as the one computable by running* $\mathsf{Witness}(\mathsf{acc}', \mathcal{X}', y, ek', \mathsf{aux}')$*.*

We point out that the ability to update membership witnesses is inherently more important than that of non-membership witnesses. The former corresponds to the (polynomially many) values in the set whereas the latter will be exponentially many (or infinite). A server that wants to cache witness values and update them efficiently can thus benefit more from storing pre-computed positive witnesses than negative ones (that are less likely to be used again).

Untrusted vs. trusted setup. The way we formulated our definition, Setup and Update require knowledge of sk, $\mathsf{Witness}$ requires ek and Verify takes only vk. From a practical point of view, the owner is the party that is responsible for maintaining the accumulation value at all times (e.g., signing it and posting it to a public log); all changes in $\mathcal{X}$ should, in a sense, be validated by him first. On the other hand, in most popular schemes (e.g., the RSA construction of [11] and the bilinear accumulator of [53]), setup and update can be run by the server (without trapdoor sk) and the only distinction is that the owner can achieve this much faster. The same holds for our construction, but in the security definitions we adopt the more general framework where the adversary is given oracle access to these algorithms. It should be noted that for our construction, all security properties hold even if sk is empty –only the complexity analysis changes.

3.1 Zero-Knowledge Accumulators: Security Properties

The first property we require from a cryptographic accumulator is completeness, i.e., a witness output by any sequence of invocations of the scheme algorithms, for a valid statement (corresponding to the state of the set at the time of the witness generation) is verified correctly with all but negligible probability.

Definition 2 (Completeness). *Let $\mathcal{X}_i$ denote the set with elements from $\mathbb{X}$, constructed after i invocations of the* Update *algorithm (starting from a set $\mathcal{X}_0$) and likewise for ek_i, aux_i. A dynamic universal accumulator is complete if, for all sets $\mathcal{X}_0$ where $|\mathcal{X}_0|$ and $l \geq 0$ are polynomial in λ and for all $x_i \in \mathbb{X}$, for $0 = 1, \ldots, l$, there exists a negligible function $\nu(\lambda)$ such that:*

$$\Pr\left[\begin{array}{c}(sk, vk) \leftarrow \mathsf{GenKey}(1^\lambda); (ek_0, \mathsf{acc}_0, \mathsf{aux}_0) \leftarrow \mathsf{Setup}(sk, \mathcal{X}_0);\\ \{(\mathsf{acc}_{i+1}, ek_{i+1}, \mathsf{aux}_{i+1}) \leftarrow \mathsf{Update}(\mathsf{acc}_i, \mathcal{X}_i, x_i, sk, \mathsf{aux}_i, \mathsf{upd}_i)\}_{0 \leq i \leq l}\\ (b, \mathsf{w}) \leftarrow \mathsf{Witness}(\mathsf{acc}_l, \mathcal{X}_l, x, ek_l, \mathsf{aux}_l) : \mathsf{Verify}(\mathsf{acc}_l, x, b, \mathsf{w}, vk) = \mathsf{accept}\end{array}\right] \geq 1 - \nu(\lambda)$$

where the probability is taken over the randomness of the algorithms.

In the above we purposely omitted the $\mathsf{WitUpdate}$ algorithm that was introduced purely for efficiency gains at the server. In fact, recall that we restricted it to return the exact same output as Update (run for the corresponding set and element) hence the value w in the above definition might as well have been computed during an earlier update and subsequently updated by (one or more) calls of $\mathsf{WitUpdate}$.

The second property is soundness which captures that fact that adversarial servers cannot provide accepting witnesses for incorrect statements. It is formulated as the inability of Adv to win a game during which he is given oracle access to all the algorithms of the scheme (except for those he can run on his own using ek, aux –see discussion on private versus public setup and updates above) and is required to output such a statement and a corresponding witness.

Definition 3 (Soundness). *For all PPT adversaries* Adv *running on input* 1^λ *and all l polynomial in λ, the probability of winning the following game, taken over the randomness of the algorithms and the coins of* Adv *is negligible:*

Setup. *The challenger runs* $(sk, vk) \leftarrow \mathsf{GenKey}(1^\lambda)$ *and forwards vk to* Adv. *The latter responds with a set $\mathcal{X}_0$. The challenger runs* $(ek_0, \mathsf{acc}_0, \mathsf{aux}_0) \leftarrow \mathsf{Setup}(sk, \mathcal{X}_0)$ *and sends the output to the adversary.*

Updates. *The challenger initiates a list $\mathcal{L}$ and inserts the tuple $(acc_0, \mathcal{X}_0)$. Following this, the adversary issues update x_i and receives the output of* $\mathsf{Update}(\mathsf{acc}_i, \mathcal{X}_i, x_i, sk, \mathsf{aux}_i, \mathsf{upd}_i)$ *from the challenger, for $i = 0, \ldots, l$. After each invocation of* Update*, if the output is not $\bot$, the challenger appends the returned $(\mathsf{acc}_{i+1}, \mathcal{X}_{i+1})$ to $\mathcal{L}$. Otherwise, he appends $(\mathsf{acc}_i, \mathcal{X}_i)$.*

Challenge. *The adversary outputs an index j, and a triplet (x^*, b^*, w^*). Let $\mathcal{L}[j]$ be $(\mathsf{acc}_j, \mathcal{X}_j)$. The adversary wins the game if:*

$$\mathsf{Verify}(\mathsf{acc}_j, x^*, b^*, \mathsf{w}^*, vk) = \mathsf{accept} \wedge ((x^* \in \mathcal{X}_j \wedge b^* = 0) \vee (x^* \notin \mathcal{X}_j \wedge b^* = 1))$$

A discussion on the winning conditions of the game is due at this point. This property (also referred to as collision-freeness) was introduced in this format in [41] and was more recently adapted in [23] with slight modifications. In particular, Adv outputs set $\mathcal{X}^*$ and accumulation value acc^* as well as the randomness used (possibly) to compute the latter (to cater for randomized accumulators). It is trivial to show that the two versions of the property are equivalent.

An alternative, more demanding, way to formulate the game is to require that the adversary wins if he outputs two accepting witnesses for the same element and with respect to the same accumulation value (without revealing the pre-image set): a membership and a non-membership one. This property, introduced in the context of accumulators in [7], is known as *undeniability* and is the same as the privacy property of zero-knowledge sets. Recently, Derler et al. [23] showed that undeniability is a stronger property than soundness. However, existing constructions for undeniable accumulators are in the trapdoor-less setting (with the limitations discussed in Sect. 1.1); since our construction is in the three-party setting, we restrict our attention to soundness. This should come as no surprise, as undeniability allows an adversary to provide a candidate accumulation value, without explicitly giving a corresponding set. In a trusted-setup setting, the accumulation value is always maintained by the trusted owner; there is no need to question whether it was honestly computed (e.g., whether he knows a set pre-image or even whether there exists one) hence undeniability in this model is an “overkill” in terms of security (see also the related discussion in Sect. 3.2).

The novel property we introduce here is zero-knowledge. Informally, this property ensures that an adversarial party (i.e., the client) that sees the accumulation value as well as all membership and non-membership witnesses exchanged during the protocol execution, and has the ability to issue arbitrary queries, learns nothing about the set, *not even its size.* Zero-knowledge guarantees that nothing can be learned from the protocol except for the answer to a query itself. In other words, explicitly querying for an element is the only way to learn whether it appears in the set or not. We formalize this in a way that is very similar to zero-knowledge sets (e.g., see the definition of [15]) appropriately extended to handle not only queries but also updates issued by the adversary. In particular, we want the proofs to be ephemeral, i.e., proofs generated before an update should be invalidated after an update. We require that there exists a simulator such that no adversarial client can distinguish whether he is interacting with the algorithms of the scheme or with the simulator that has no knowledge of the set or the element updates that occur, other than whether a queried element is in the set and whether requested updates are valid. This information is given to the simulator as the output of a function D that checks the validity of a requested operation[5].

Definition 4 (Zero-Knowledge). *Let D be a binary function defined as follows. For queries, $D(\mathsf{query}, x, \mathcal{X})) = 1$ iff $x \in \mathcal{X}$. For updates $D(\mathsf{update}, x, c, \mathcal{X})) = 1$ iff $(c = 1 \wedge x \notin \mathcal{X})$ or $(c = 0 \wedge x \in \mathcal{X})$. Let $\mathsf{Real}_{\mathsf{Adv}}(1^\lambda), \mathsf{Ideal}_{\mathsf{Adv},\mathsf{Sim}}(1^\lambda)$ be games between a challenger, an adversary* Adv *and a simulator* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$*, defined as follows:*

$\mathsf{Real}_{\mathsf{Adv}}(1^\lambda)$:

- **Setup.** *The challenger runs $(sk, vk) \leftarrow \mathsf{GenKey}(1^\lambda)$ and forwards vk to* Adv*. The latter chooses a set $\mathcal{X}_0$ with $|\mathcal{X}_0| \in poly(\lambda)$ and sends it to the challenger who in turn runs $\mathsf{Setup}(sk, \mathcal{X}_0)$ to get $(\mathsf{acc}_0, ek_0, \mathsf{aux}_0)$. He then sends acc_0 to* Adv *and sets $(\mathcal{X}, \mathsf{acc}, ek, \mathsf{aux}) \leftarrow (\mathcal{X}_0, \mathsf{acc}_0, ek_0, \mathsf{aux}_0)$.*
- **Query.** *For $i = 1, \ldots, l$, where $l \in poly(\lambda)$,* Adv *outputs (op, x_i, c_i) where $\mathsf{op} \in \{\mathsf{query}, \mathsf{update}\}$ and $c_i \in \{0, 1\}$:*
  - **If** $\mathsf{op} = \mathsf{query}$*: The challenger runs $(b, \mathsf{w}_i) \leftarrow \mathsf{Witness}(\mathsf{acc}, \mathcal{X}, x_i, ek, \mathsf{aux})$ and returns the output to* Adv*.*
  - **If** $\mathsf{op} = \mathsf{update}$*: The challenger runs $\mathsf{Update}(\mathsf{acc}, \mathcal{X}, x_i, sk, \mathsf{aux}, c_i)$. If the output is not $\perp$ he updates the set accordingly to get $\mathcal{X}_i$, sets $(\mathcal{X}, \mathsf{acc}, ek, \mathsf{aux}) \leftarrow (\mathcal{X}_i, \mathsf{acc}_i, ek_i, \mathsf{aux}_i)$ and forwards* acc *to* Adv*. Else, he responds with $\perp$.*
- **Response.** *The adversary outputs a bit d.*

$\mathsf{Ideal}_{\mathsf{Adv}}(1^\lambda)$:

- **Setup.** *The simulator Sim_1, on input 1^λ, outputs a vk which he forwards to* Adv*. The adversary chooses a set $\mathcal{X}_0$ with $|\mathcal{X}_0| \in poly(\lambda)$. Sim_1 (without seeing $\mathcal{X}_0$) responds with acc_0 and maintains state $\mathsf{state}_{\mathcal{S}}$. Finally, let $(\mathcal{X}, \mathsf{acc}) \leftarrow (\mathcal{X}_0, \mathsf{acc}_0)$.*
- **Query.** *For $i = 1, \ldots, l$* Adv *outputs (op, x_i, c_i) where $\mathsf{op} \in \{\mathsf{query}, \mathsf{update}\}$ and $c_i \in \{0, 1\}$:*
  - **If** $\mathsf{op} = \mathsf{query}$*: The simulator runs $(b, \mathsf{w}_i) \leftarrow \mathsf{Sim}_2(\mathsf{acc}, x_i, \mathsf{state}_{\mathcal{S}}, D(\mathsf{query}, x_i, \mathcal{X}))$ and returns the output to* Adv*.*
  - **If** $\mathsf{op} = \mathsf{update}$*: The simulator runs $\mathsf{Sim}_2(\mathsf{acc}, \mathsf{state}_{\mathcal{S}}, D(\mathsf{update}, x_i, c_i, \mathcal{X}))$. If the output of $D(\mathsf{update}, x_i, c_i, \mathcal{X})$ is 1, let $\mathcal{X} \leftarrow \mathcal{X}_i \cup x_i$ in the case $c_1 = 1$ and $\mathcal{X} \leftarrow \mathcal{X}_i \setminus x_i$ in the case $c_1 = 0$ —i.e., $\mathcal{X}$ is a placeholder variable for the latest set version at all times according to valid updates, that is however never observed by the simulator. The simulator responds to* Adv *with* acc'*. If the response acc' is not $\perp$ then $\mathsf{acc} \leftarrow \mathsf{acc}'$.*

[5] Instead of using D with different arguments for checking the validity of query and update, we could make D work only for queries, i.e., $D(\mathsf{query}, \ldots)$, and express the validity of a requested update as $D(\mathsf{query}, \ldots) \oplus c_i$. We chose to use the former notation because we feel it is cleaner.

Response. *The adversary outputs a bit d.*

A dynamic universal accumulator is zero-knowledge if there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that for all adversaries* Adv *there exists negligible function* ν *such that:*

$$|\Pr[\mathsf{Real}_{\mathsf{Adv}}(1^\lambda) = 1] - \Pr[\mathsf{Ideal}_{\mathsf{Adv}}(1^\lambda) = 1]| \leq \nu(\lambda).$$

If Adv *is PPT, then this defines computational zero-knowledge; perfect and statistical zero-knowledge can be defined similarly.*

Observe that, even though Adv may be unbounded (in the case of statistical or perfect zero-knowledge) the size of the set is always polynomial in the security parameter as in [15]; in fact it is upper bounded by $|\mathcal{X}_0| + l$. This ensures that we can have polynomial-time simulation, matching the real-world execution where all parties run in polynomial-time. Having computationally unbounded adversaries is still meaningful; such a party may, after having requested polynomially many updates, spend unlimited computational resources trying to distinguish the two settings.

As already observed in [20,21,23], when formulating a notion of privacy for cryptographic accumulators the fact that the accumulation value computation must be randomized becomes evident. If Setup (and similarly, Update) is a deterministic algorithm, then each set has a uniquely defined accumulation value (subject to particular sk) that can be reproduced by any adversary with oracle access to the algorithm.

In our definition, the server holds the evaluation key ek that is used to produce witnesses, and that is not available to the client. This is not a restriction of the model, but should rather be seen as a generalization, in order to capture zero-knowledge in all settings; if ek does not leak any information about the set, it may be included in the public vk. Specifically for our construction from Sect. 4, if we choose to make ek public, then what is leaked is an upper-bound on the set size, formally captured by the notion of *functional zero-knowledge* [52].

3.2 Relation to Other Primitives

There exist various cryptographic primitives that address the problem of secure set (non-)membership, in the same or related models, and it is imperative to compare the primitive of zero-knowledge accumulators with these.

We present a mapping of the research literature for the construction of cryptographic proofs for set-membership and non-membership, which has attracted significant attention lately; proofs can be found in the full version. This is far from a complete presentation of results in the area; we focus on the relation between those primitives that are most closely related to the problem, avoiding general approaches (e.g., general-purpose zero-knowledge protocols) or related models that address similar problems (such as group signatures, e.g., [1]). The overall picture for the static case (i.e., without assuming changes in the set) can be seen in Fig. 1. Arrows denote implication; an arrow from A to B translates to "B can be built in black-box manner from A". Double-sided arrows denote equivalence of definitions, i.e., both can be constructed in a black-box manner from each other.

The most prominent such primitive is zero-knowledge sets [14,15,43,48]. There, queries can be answered without revealing anything about the set, albeit at a stronger setting where the server and the owner are the same (untrusted) entity. In the same setting, we also discussed trapdoorless (or strong) accumulators (see Sect. 1.1). Zero-knowledge sets are a stronger primitive than accumulators; they satisfy the same soundness property with trapdoorless accumulators but they additionally offer privacy. Hence all other primitives in our mapping can be built from them. Additionally, if a scheme is a trapdoorless accumulator it is secure with an untrusted setup execution, therefore (and quite trivially) it is also secure with a trusted setup, hence it is a also an accumulator.

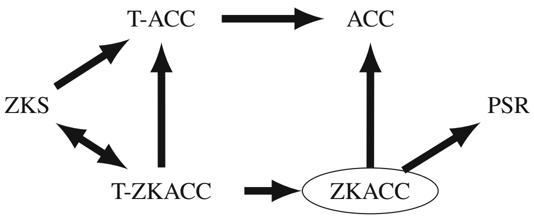

Fig. 1. Relations among cryptographic primitives for proof of membership and non-membership (static case). ZKS: zero-knowledge sets, T-ACC: trapdoorless accumulators, ACC: accumulators, T-ZKACC: trapdoorless zero-knowledge accumulators, ZKACC: our zero-knowledge accumulators (circled), PSR: primary-secondary-resolver membership proof systems.

As a mental exercise, let us now try to define the privacy-preserving counterparts of trapdoorless accumulators, i.e., *trapdoorless zero-knowledge accumulators*[6]. Quite informally, the completeness and zero-knowledge definitions remain the same but the soundness property is replaced by the, strictly stronger, property of undeniability (see, e.g., [44] for a concrete definition), which is the same as the soundness property of zero-knowledge sets: By "merging" the existing soundness guarantee of trapdoorless accumulators with our zero-knowledge property (which, for the static case, is identical to that of zero-knowledge sets) we –quite unsurprisingly– ended up with zero-knowledge sets. We stress that the latter exist in the common reference string model (or the trusted parameters model) hence this must also be true for trapdoorless zero-knowledge accumulators (e.g., a trusted authority runs the key-generation algorithm and publishes the result as a common reference string). On the contrary, this is not necessary for trapdoorless accumulators (without privacy) since the security game there is one-sided; the client can perform key-generation himself. As a final note, we point out, that zero knowledge (trapdoorless) accumulators

[6] It should be noted that, in the accumulators literature, the trapdoor refers to a secret value possibly used for efficiency purposes when computing accumulation values and witnesses by the trusted owner. This should not be confused with the trapdoor typically used in zero-knowledge protocols for simulation purposes.

imply (trapdoorless) accumulators since the former satisfy a strict superset of the security properties of the latter.

This equivalence of zero-knowledge sets and trapdoorless zero-knowledge accumulators can be useful in two ways: (i) more efficient (e.g., with smaller proof sizes) zero-knowledge sets may be achievable with techniques borrowed from the accumulators literature, and (ii) an impossibility result in one of the two models is translatable to the other. This holds, for example, in the case of the batch-update impossibility for accumulators of [8]. We want to stress that our construction in Sect. 4 is not trapdoorless; to the best of our knowledge, the best known way to construct trapdoorless zero-knowledge accumulators is via a black-box reduction from zero-knowledge sets.

Another related primitive is primary-secondary-resolver proof systems (PSR), introduced by Naor et al. [52]. Their privacy notion is a relaxation of zero-knowledge defined as *functional zero-knowledge*, i.e., the simulator may be allowed to learn some function of the set (typically its size). Also, the games in the PSR definition are non-adaptive in the following sense: Adv needs to declare its cheating set before he even receives the corresponding keys (ek, vk for soundness and only vk for zero-knowledge –using our terminology)[7]. For the above reasons, while it is trivial that zero-knowledge accumulators imply PSR (where the leaked function is void), the other direction is generally not true. We stress that the above distinction between adaptive and selective security does not hold in the dynamic setting. There an adversary may declare a cheating set originally, receive the keys, and then modify his choice via a series of update calls (see, however, our discussion for this setting in the next paragraph).

Our results here are complementary to the relations proven in [52]. There, the authors prove that PSR systems exist, if and only if, one-way functions exist, which in turn implies that zero-knowledge sets cannot be built in a black-box manner from PSR.

Dynamic setting. Once we move to the dynamic setting, where there exist efficient algorithms for modifications in the set, the relations are largely the same as in Fig. 1, but some clarifications are in order. The first work addressing updatable zero-knowledge sets was by Liskov [45], where two notions of privacy were introduced: opacity and transparency. Constructions of the latter form were presented in [13,45]. The above relations between definitions hold with respect to opacity. A construction for efficiently updatable opaque zero-knowledge sets (from standard assumptions) remains an open problem. However, when restricted to the three-party model (i.e., with trusted setup), it can be shown that our construction from Sect. 4 (with minor modifications) satisfies the opacity property. On the other hand, transparency is a weaker form of privacy, as it trivially leaks

[7] One could possibly modify the PSR model –and the security games– significantly to make them adaptive, by separating the key generation and setup algorithms. Indeed, to the best of our knowledge, the PSR construction of [33] would probably satisfy such a modified definition, assuming it was instantiated with an adaptively-secure signature scheme and an adaptively-secure verifiable random function.

whether a particular element, that has been previously queried, was affected by an update (but it otherwise allows parties to maintain cached witnesses).

Regarding the relation between zero-knowledge accumulators and PSR, matters are also straight-forward as the latter are explicitly defined only for the static case. In [52], the authors recommend the usage of techniques from certificate-revocation lists [51], as an additional external mechanism to accommodate updates. Contrary to this, our definitional approach is to make update-handling mechanisms explicitly part of the scheme. In this sense, zero-knowledge accumulators are a natural definitional extension of PSR in the dynamic setting. That said, we explicitly require that clients can at all times access the latest accumulation value, which would not be the case following the revocation scheme approach. We stress however that this does not necessitate active authenticated channels between owner and clients; in practice it is achievable with a "timestamp-sign-and-publish" from the owner.

We note that recently [42] introduced the general notion of functional commitments (which can capture accumulators as a special case). However, their construction handles only subset queries and it does not support updates on the committed set. On the other hand, [60] introduced the notion of asynchronous accumulators in a distributed setting and does not consider privacy.

Relation to zero-knowledge authenticated data structures. Another important observation is the relation of zero-knowledge accumulators with the framework of zero-knowledge authenticated data structures (ZKADS), recently introduced in [29].[8] ZKADS extend the well-known primitive of authenticated data structures (ADS) adding an additional zero-knowledge property. The setting is the standard three-party model but now the supported type may be any kind of data structure. The choice of data structure defines the kind of data stored and the type of supported queries. In [29,31,32], the authors provided constructions for various types of data structures, in particular for a zero-knowledge authenticated list (i.e., a data structure that supports "insert-after", "delete" operations, as well as "order" queries), a tree, and a partially-ordered set (poset) of bounded dimension and range queries. Consequently, a zero-knowledge accumulator is a type of ZKADS where the data structure is a set of elements supporting –unordered– insertions and deletions, and (non-)membership queries.

The above constructions are the only ZKADS instantiations in the literature so far. One natural way to extend zero-knowledge authenticated sets to accommodate more elaborate query types is by allowing for set-operations beyond (non-)membership. In particular, consider a data structure, called set collection, that consists of a collection of sets and accommodates operations among (an arbitrary selection among) them. We stress that a construction that accommodates set unions, intersection and differences, allows for a complete set algebra.[9] In the

[8] Though [29] uses the term Privacy-Preserving Authenticated Data Structures, we use ZKADS to fit our notation.

[9] In the computationally-bounded setting, a negation operation is infeasible unless the element domain is of polynomial size in the security parameter. In that case, a negation can be instantiated as a set difference from the set that contains the entire domain.

full version we provide a definition of *zero-knowledge authenticated set collection*, in the style of [29], and the corresponding construction (which naturally uses our zero-knowledge authenticated set construction from Sect. 4 as a building block).

3.3 Zero-Knowledge Implies Indistinguishability (for Accumulators)

The notion of zero-knowledge defined here is a strengthening of the indistinguishability property introduced in [23]. There the authors introduce a notion similar to ours that also requires the accumulation value produced by Setup to be randomized. If we restrict our attention to static accumulators, the effect of both notions is the same, i.e., the clients see a randomized accumulation value and corresponding "blinded" witnesses.

However, while the indistiguishability game entails updates, it inherently does not offer any privacy for the elements inserted to or removed from the set, as the Update algorithm is deterministic. At a high level, that notion only protects the original accumulated set and not subsequent updates. We believe this is an important omission for a meaningful privacy definition for accumulators, as highlighted by the following example. Consider, a third-party adversary that observes the protocol's execution before and after an insertion (or deletion) update. If the adversary has reasons to suspect that the inserted (or deleted) value may be y, he can always test that. A very realistic example of this behavior is a setting where the accumulator is used to implement a revocation list. In that case an adversary may want to know if his fake certificate (value y in the above case) has been "caught" yet. We provide the following result[10].

Theorem 1. *Every zero-knowledge dynamic universal accumulator is also indistinguishable under the definition of [23], while the opposite is not always true.*

Proof. We first show that every scheme that is zero-knowledge is also indistinguishable. Then we show that the construction of [23] is not zero-knowledge.

ZK ⇒ IND: We prove this direction by contradiction. Assume there exists an accumulator that is zero-knowledge but not indistinguishable. Then, there exists a PPT adversary Adv that wins the indistinguishability game. Adv gives two sets $\mathcal{X}_0, \mathcal{X}_1$ to a challenger who flips a coin b and provides oracle access to Adv for the algorithms with respect to $\mathcal{X}_b$. By assumption, Adv can

[10] In [23] the indistinguishability definition assumes that the adversary is also given access to the Setup algorithm arbitrarily many times. This makes sense in their model, since they explicitly require that Setup is randomized whereas Update is deterministic. Here this requirement is redundant since both processes may be randomized; any setup response can be emulated by a series of update calls that shape the required set. To simplify the process, we assume that the indistinguishability adversary only makes Update and $\mathsf{Witness}$ calls. We stress that this is not a limitation of the reduction. We could alternatively have chosen to define our zero-knowledge game giving the adversary access to Setup and the result would still hold.

output a bit b' correctly guessing b with non-negligible advantage ϵ over $1/2$. The (natural) constraint is that Adv cannot issue a query (or update request) that is trivially revealing the chosen set (e.g., if $x \in \mathcal{X}_0$ and $x \notin \mathcal{X}_1$, Adv is not allowed to query for x). We defer interested readers to [23] for a formal definition of the indistinguishability game.

We will now construct a PPT adversary Adv' that breaks the zero-knowledge property of the scheme as follows. Adv' on input $1^\lambda, vk$ runs Adv with the same input and receives sets $\mathcal{X}_0, \mathcal{X}_1$. He then forwards $\mathcal{X}_1$ as the challenge for the zero-knowledge game and receives accumulation value acc_0, which he forwards to Adv. Consequently, he responds to all messages of Adv (queries and updates) with calls to the zero-knowledge game interface and forwards all responses back to Adv. Finally, he outputs the output bit b' of Adv.

First, observe that Adv' is clearly PPT, since Adv is PPT. Now let us argue about his success probability in distinguishing between real and ideal interaction. Observe that, if Adv' is interacting with the algorithms of the scheme (i.e., is playing the real game), the interface he is providing to Adv is a perfect simulation of the indistinguishability game for $b = 1$. On the other hand, if he is interacting with Sim, the view of the latter during this interaction is exactly the same independently of whether the set chosen by Adv' is $\mathcal{X}_0$ or $\mathcal{X}_1$. Hence, the view offered to Adv is the same in both cases, and therefore $\Pr[b' = 1] = \Pr[b' = 0] = 1/2$. Let E be the event that the Adv' is playing the real game (and likewise for the complement E^c). From the above analysis (recall that Adv' outputs the bit b' returned by Adv), it holds that $\Pr[b' = 1|E] > 1/2 + \epsilon$ and $\Pr[b' = 1|E^c] = 1/2$. This implies that Adv' can distinguish between the two executions with non-negligible probability, breaking the zero-knowledge property of the scheme. The claim follows by contradiction.

IND $\not\Rightarrow$ ZK: The main observation for this part of the proof is that in the construction of [23], given the accumulation acc of set $\mathcal{X}$, the new accumulation value after inserting or deleting an element is computed via a deterministic algorithm. Assume now an adversary Adv that operates as follows when playing the zero-knowledge game against the scheme of [23]. He initially plays the setup phase of Definition 4 choosing a set $\mathcal{X}_0$ and receiving acc_0 from the challenger. Then he chooses $e \xleftarrow{\$} \{0, 1\}$. If $e = 0$ then Adv chooses x uniformly from $\mathbb{Z}_p \setminus \{\mathcal{X}_0\}$ and sends to the challenger first $(\mathsf{update}, x, 1)$, receiving acc_1, and then $(\mathsf{update}, x, 0)$, receiving acc_2. Else, if $e = 1$ he chooses x, y uniformly from $\mathbb{Z}_p \setminus \{\mathcal{X}_0\}$ with $y \neq x$, and sends to the challenger first $(\mathsf{update}, x, 1)$, receiving acc_1, and then $(\mathsf{update}, y, 1)$, receiving acc_2. Finally, if $(\mathsf{acc}_2 = \mathsf{acc}_0 \wedge e = 0)$ or $(\mathsf{acc}_2 \neq \mathsf{acc}_0 \wedge e = 1)$, he outputs $d = 1$. In all other cases he outputs $d = 0$.

Observe first that Adv is clearly PPT as all algorithms of the scheme are run in polynomial time. Regarding his success probability, we argue as follows. If Adv is playing the real game versus the challenger running the algorithms of [23], then we identify the following two probabilities $\Pr[\mathsf{acc}_2 = \mathsf{acc}_0 | e = 0]$ and $\Pr[\mathsf{acc}_2 = \mathsf{acc}_0 | e = 1]$. The first probability is equal to 1 whereas the second one is negligibly small; as explained above, the updates of the scheme are deterministic therefore adding and removing the same element will result in the same accumulation value, whereas adding two elements will always result in a different accumulation value, unless the latter happens to be the multiplicative inverse of the former. On the other hand, if Adv is playing the ideal game against the simulator, the latter is only given access to the information that two updates occurred (not even the nature of the update operations). Therefore, the simulator's view is the same, independently of the value of e, and $\Pr[\mathsf{acc}_2 = \mathsf{acc}_0 | e = 0] = \Pr[\mathsf{acc}_2 = \mathsf{acc}_0 | e = 1] = 1/2$. Let E be the event that the Adv' is playing the real game (and likewise for the complement E^c). From the above analysis it follows that $Pr[d = 1|E] = 1 - \nu(\lambda)$, whereas $Pr[d = 1|E^c] = 1/2$ therefore Adv distinguishes the two games with non-negligible probability and the accumulator of [23] is not zero-knowledge. ∎

Other privacy notions. The indistinguishability property of [23] is a strengthening of a that of [20]. The latter was the first work to formally define a privacy property for cryptographic accumulators, however their definition had inherent problems, e.g., it was easy to prove that deterministic accumulators –that clearly were not private– satisfied it. Another technique for providing privacy to cryptographic accumulators was proposed earlier in [41], without a formalization. The idea is to simply produce a randomized accumulation value for a set $\mathcal{X}$ by choosing at random an element x from the elements universe during Setup and outputting the accumulation of set $\mathcal{X} \cup \{x\}$. This generic mechanism will work for any static accumulator, but will also not protect updates. Moreover it weakens soundness as an adversary could potentially produce a membership witness for the element $x \notin \mathcal{X}$. Out approach does not suffer from this as there is no additional element accumulated and the randomness r used to blind the accumulation value during Setup is explicitly given to the server without compromising soundness. Finally, Theorem 1 implies that our construction from Sect. 4, is also the only known algebraic construction of a universal indistinguishable accumulator. The two schemes of [23] are a black-box reduction from the stronger primitive of zero-knowledge sets, and a construction similar to ours that only offers membership witnesses.

4 A Zero-Knowledge Universal Accumulator Construction

In this section we present our construction for a zero-knowledge dynamic universal accumulator. It builds upon the bilinear accumulator of Nguyen [53], adopting some of the techniques of [23] that we further expand to achieve zero-knowledge. It supports sets with elements from $\mathbb{Z}_p \setminus \{s\}$ where p is prime and $p \in O(2^\lambda)$ and s is the scheme trapdoor. Note that, the fact that the elements must be of $\log p$ bits each, is not a strong limitation of the scheme; one can always apply a collision-resistant hash function that maps arbitrarily long strings to $\mathbb{Z}_p$. We now make several observations about our ZKUA construction in Fig. 2.

Notation: The notation $q[z]$ denotes polynomial q over undefined variable z and $q(s)$ is the evaluation of the polynomial at point s. All arithmetic operations are performed $\bmod p$. N is a variable maintained by the owner.

Key Generation $(sk, vk) \leftarrow \mathsf{GenKey}(1^{\lambda})$

Run $\mathsf{GenParams}(1^k)$ to receive bilinear parameters $pub = (p, \mathbb{G}, \mathbb{G}_T, e, g)$. Choose $s \xleftarrow{\$} \mathbb{Z}_p^*$. Return $sk = s$ and $vk = (g^s, pub)$.

Setup $(\mathsf{acc}, ek, \mathsf{aux}) \leftarrow \mathsf{Setup}(sk, \mathcal{X})$

Choose $r \xleftarrow{\$} \mathbb{Z}_p^*$. Set value $N = |\mathcal{X}|$. Return $\mathsf{acc} = g^{r\cdot \mathsf{Ch}_{\mathcal{X}}(s)}$, $ek = (g, g^s, g^{s^2}, \dots, g^{s^N})$ and $\mathsf{aux} = (r, N)$.

Witness Generation $(b, \mathsf{w}) \leftarrow \mathsf{Witness}(\mathsf{acc}, \mathcal{X}, x, ek, \mathsf{aux})$

If $x \in \mathcal{X}$ compute $\mathsf{w} = (\mathsf{acc})^{\frac{1}{s+x}} = g^{r\cdot \mathsf{Ch}_{\mathcal{X}\setminus\{x\}}(s)}$ and return $(1, \mathsf{w})$.

Else, proceed as follows:

- Using the Extended Euclidean algorithm, compute polynomials $q_1[z], q_2[z]$ such that $q_1[z]\mathsf{Ch}_{\mathcal{X}}[z] + q_2[z]\mathsf{Ch}_{\{x\}}[z] = 1$.
- Pick a random $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$ and set $q'_1[z] = q_1[z] + \gamma \cdot \mathsf{Ch}_{\{x\}}[z]$ and $q'_2[z] = q_2[z] - \gamma \cdot \mathsf{Ch}_{\mathcal{X}}[z]$.
- Set $W_1 := g^{q'_1(s)r^{-1}}, W_2 = g^{q'_2(s)}$ and $\mathsf{w} := (W_1, W_2)$. Return $(0, \mathsf{w})$.

Verification $(\mathsf{accept/reject}) \leftarrow \mathsf{Verify}(\mathsf{acc}, x, b, \mathsf{w}, vk)$

If $b = 1$ return accept if $e(\mathsf{acc}, g) = e(\mathsf{w}, g^x \cdot g^s)$, reject otherwise. If $b = 0$ do the following:

- Parse w as (W_1, W_2).
- Return accept if $e(W_1, \mathsf{acc})e(W_2, g^x \cdot g^s) = e(g, g)$, reject otherwise.

Update $(\mathsf{acc}', ek', \mathsf{aux}') \leftarrow \mathsf{Update}(\mathsf{acc}, \mathcal{X}, x, sk, \mathsf{aux}, \mathsf{upd})$

Parse aux as (r, N). If $(\mathsf{upd} = 1 \wedge x \in \mathcal{X})$ or $(\mathsf{upd} = 0 \wedge x \notin \mathcal{X})$ output $\perp$ and halt. Choose $r' \xleftarrow{\$} \mathbb{Z}_p^*$. If $\mathsf{upd} = 1$:

- Compute $\mathsf{acc}' = \mathsf{acc}^{(s+x)r'}$.
- If $|\mathcal{X}| + 1 > N$, set $N = |\mathcal{X}| + 1$ and compute $\mathsf{ek}' = g^{s^N}$.

Else, compute $\mathsf{acc}' = \mathsf{acc}^{\frac{r'}{s+x}}$ and $ek' = \emptyset$. In both cases, set $\mathsf{aux}' := (r \cdot r', N)$ and return $(\mathsf{acc}', ek', \mathsf{aux}')$.

Witness Update $(\mathsf{upd}, \mathsf{w}') \leftarrow \mathsf{WitUpdate}(\mathsf{acc}, \mathsf{acc}', x, \mathsf{w}, y, ek', \mathsf{aux}, \mathsf{aux}', \mathsf{upd})$

Parse $\mathsf{aux}, \mathsf{aux}'$ to get r, r'.

- If w is a membership witness:

  If $\mathsf{upd} = 1$ output $(1, \mathsf{w}' = (\mathsf{acc} \cdot \mathsf{w}^{x-y})^{r'})$. Else, output $(0, \mathsf{w}' = \mathsf{acc}'^{\frac{1}{(y-x)}} \cdot \mathsf{w}^{\frac{r'}{(x-y)}})$.
- If w is a non-membership witness:

  Let $\mathcal{X}'$ be the set produced after the execution of Update for element x (i.e., the current set). Run $\mathsf{Witness}(\mathsf{acc}', \mathcal{X}', y, ek', \mathsf{aux}')$ and return its output.

Fig. 2. Zero-knowledge dynamic universal accumulator construction.

The main property of our construction is that the algorithms do not reveal anything about the set in the units sent to the client. The key vk published from the key-generating algorithm reveals nothing about the set. The accumulation value produced by Setup is the standard bilinear accumulation value of [53] which is now blinded by a random value r, also revealed to the server. Witness generation also utilizes this randomness r.

For membership queries, the process is the same as in [19,53] with one additional exponentiation with r for privacy purposes. This technique proves that an element $x \in \mathcal{X}$ iff the degree-one polynomial $x + z$ divides $\mathsf{Ch}_{\mathcal{X}}[z]$. The major deviation occurs in the non-membership case. As previously discussed, there are existing works [2,19] that enhance the bilinear accumulator to provide non-membership witnesses. Their technique is a complement of the one used for the membership case. At a high level, it entails proving that the degree-one polynomial $x + z$ does not divide $\mathsf{Ch}_{\mathcal{X}}[z]$, by revealing the scalar (i.e., zero-degree polynomial) remainder of their long division. Unfortunately, using this approach here entirely breaks the zero-knowledge property: It reveals r (multiplied by an easily computable query-specific value) to any client. Instead, we adopt an entirely different approach. Our scheme uses the set-disjointness test, first proposed in [56]. In order to prove that $x \notin \mathcal{X}$, the server proves the statement $\mathcal{X} \cap \{x\} = \emptyset$. The different nature of the proved statement allows us to use fresh query-specific randomness γ together with r to prove non-membership in zero-knowledge. As a consequence, the verification for membership and non-membership is also different, but always efficient.

Finally, the way updates are handled is especially important as it is another strong point of divergence from previous schemes that seek to provide privacy. After each update, a fresh randomness r' is used to blind the new accumulation value. This re-randomization technique perfectly hides the nature of the change in $\mathcal{X}$ and lets us achieve zero-knowledge. Observe that, at all times, the owner maintains a variable N which is the maximum set-cardinality observed up to that point (through the original setup and subsequent insertions). If an insertion increases N (at most by one), the owner provides the server with an additional ek component that can be used by the server for subsequent witness generation. This is a slight deviation from our notation in Sect. 3 where the new key produced from Update replaces the previous ek. Instead the new evaluation key must be set to $ek \cup ek'$. This has no meaningful impact to the security of our scheme; we could always have Update output the entire old key together with the additional element. From an efficiency perspective though, that overly naive approach would require Update to run in time linear to N –the same holds for $\mathsf{WitUpdate}$. Regarding witness updates, for the (more meaningful, as discussed in Sect. 3) case of membership witnesses there indeed exists a fast method. On the other hand, for non-membership witness updates, our scheme resorts to re-computation from scratch.

We can now present our main result. We give the proof of security below and defer the asymptotic analysis to the full version [30].

Theorem 2. *The algorithms* $\{\mathsf{KeyGen}, \mathsf{Setup}, \mathsf{Witness}, \mathsf{Verify}, \mathsf{Update}, \mathsf{WitUpdate}\}$ *constitute a zero-knowledge dynamic universal accumulator with perfect completeness, soundness under the q-SBDH assumption (with $q = N$ set to the maximum set size observed during the soundness game) and perfect zero-knowledge. Let N be the cardinality of the set. Then, the runtime* *of* GenKey *is* $O(poly(\lambda))$ *where* λ *is the security parameter, the complexity of* Setup *is* $O(N)$*, that of* $\mathsf{Witness}$ *is* $O(N \log N)$ *for membership witnesses and* $O(N \log^2 N \log\log N)$ *for non-membership witnesses, that of* Verify *is* $O(1)$*, that of* Update *is* $O(1)$*, and that of* $\mathsf{WitUpdate}$ *is* $O(1)$ *for membership witnesses and* $O(N \log^2 N \log\log N)$ *for non-membership witnesses. Finally, witnesses consist of* $O(1)$ *bilinear group elements.*

Proof. Completeness follows by close inspection of the algorithms' execution. We proceed to prove soundness and zero-knowledge.

Proof of Soundness. Assume there exists PPT adversary Adv that on input 1^λ breaks the soundness of our scheme with non-negligible probability. We will construct a PPT adversary Adv' that breaks the q-SBDH assumption for $q = N$, running as follows:

1. On input $(pub, (g^s, \ldots, g^{s^N}))$, run Adv on input $(g^s, pub, 1^\lambda)$.
2. Upon receiving set $\mathcal{X}_0$, choose $r_0 \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. Use r_0 and $(g^s, \ldots, g^{s^N})$ to compute $\mathsf{acc}_0 = g^{r_0 \cdot \mathsf{Ch}_{\mathcal{X}_0}(s)} = g^{(\mathsf{Ch}_{\mathcal{X}_0}(s))^{r_0}}$ and respond with $(ek_0 = (g, g^s, \ldots, g^{s^{|\mathcal{X}_0|}}), \mathsf{acc}_0, r_0)$. Initiate list $\mathcal{L}$ and insert triplet $(\mathsf{acc}_0, \mathcal{X}_0, r_0)$ as $\mathcal{L}[0]$ (i.e., the first element of the list). The notation $\mathcal{L}[i]_j$ denotes the first part of the i-th element of the list (e.g., $\mathcal{L}[0]_0 = \mathsf{acc}_0$). Also set $n = |\mathcal{X}_0|$.
3. Initiate update counter $i = 0$. While $i \leq l$ proceed as follows. Upon receiving update upd_i, x_i, check whether this is a valid update for $\mathcal{X}_i = \mathcal{L}[i]_1$. If it is not, respond with $\perp$ and re-append $acc_i = \mathcal{L}[i]_0, \mathcal{X}_i, r_i$ to $\mathcal{L}$. Otherwise, pick $r' \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and set $r_{i+1} = r_i \cdot r'$. Update $\mathcal{X}_i$ according to upd_i, x_i to get $\mathcal{X}_{i+1}$. If $|\mathcal{X}_{i+1}| > n$, set $n = |\mathcal{X}_{i+1}|$ and $ek_{i+1} = g^{s^n}$. Else, $ek_{i+1} = \emptyset$. Use r_{i+1} and $(g^s, \ldots, g^{s^N})$ to compute $\mathsf{acc}_{i+1} = g^{r_{i+1} \cdot \mathsf{Ch}_{\mathcal{X}_{i+1}}(s)} = g^{(\mathsf{Ch}_{\mathcal{X}_{i+1}}(s))^{r_{i+1}}}$ and respond with $(ek_{i+1}, \mathsf{acc}_{i+1}, r_{i+1})$. Append triplet $(\mathsf{acc}_{i+1}, \mathcal{X}_{i+1}, r_{i+1})$ to $\mathcal{L}$. In both cases, increase i by 1.
4. Upon receiving the j-th challenge with triplet (x^*, b^*, w^*) proceed as follows:
   - If $b^* = 1$, then $x^* \notin \mathcal{X}_j$ yet $\mathsf{Verify}(\mathsf{acc}_j, x^*, 1, vk)$ accepts. Compute polynomial $q[z]$ and scalar c such that $\mathsf{Ch}_{\mathcal{X}_j}[z] = (x^* + z)q[z] + c$. Output $[x^*, (e(\mathsf{w}^*, g)^{r_j^{-1}} e(g, g^{-q(s)}))^{c^{-1}}]$.
   - If $b^* = 0$, then $x^* \in \mathcal{X}_j$ yet $\mathsf{Verify}(\mathsf{acc}_j, x^*, 0, vk)$ accepts. Parse w^* as (W_1^*, W_2^*). Compute polynomial $q[z]$ such that $\mathsf{Ch}_{\mathcal{X}_j}[z] = (x^* + z)q[z]$. Output $[x^*, (e(W_1^*, g^{r_j \cdot q(s)})e(W_2^*, g))]$.

First of all observe that Adv' perfectly emulates the challenger for the DUA security game to Adv. This holds since all accumulation values and witness are computable without access to trapdoor sk in polynomial time. All the necessary polynomial arithmetic can be also run efficiently hence Adv' is PPT. Regarding its success probability, we argue for the two cases separately as follows:

$\mathbf{b^* = 1}$ Since $x^* \notin \mathcal{X}_j$, it follows that $(x^* + z) \not| \mathsf{Ch}_{\mathcal{X}_j}[z]$ which guarantees the existence of $q[z], c$. Also observe that c is a scalar (zero-degree polynomial) since it is the remainder of the polynomial division and it must have degree less than that of $(x^* + z)$. Since verify accepts we can write

$$e(\mathsf{w}^*, g^{x^*} \cdot g^s) = e(\mathsf{w}^*, g)^{(x^*+s)} = e(\mathsf{acc}_j, g) = e(g^{r_j \cdot \mathsf{Ch}_{\mathcal{X}_j}(s)}, g) = e(g, g)^{r_j((x^*+s)q(s)+c)}$$

from which it follows that:

$$\begin{aligned} e(\mathsf{w}^*, g)^{r_j{}^{-1}(x^*+s)} &= e(g, g)^{(x^*+s)q(s)+c} \\ e(\mathsf{w}^*, g)^{r_j{}^{-1}} &= e(g, g)^{q(s)+c/(x^*+s)} \\ e(\mathsf{w}^*, g)^{r_j{}^{-1}} e(g, g)^{-q(s)} &= e(g, g)^{c/(x^*+s)} \\ [e(\mathsf{w}^*, g)^{r_j{}^{-1}} e(g, g)^{-q(s)}]^{c^{-1}} &= e(g, g)^{1/(x^*+s)}. \end{aligned}$$

$\mathbf{b}^* = \mathbf{0}$ Since $x^* \in \mathcal{X}_j$, it follows that $(x^* + z)|\mathsf{Ch}_{\mathcal{X}_j}[z]$ which guarantees the existence of $q[z]$. Since verify accepts we can write:

$$\begin{aligned} e(W_1^*, \mathsf{acc}_j)e(W_2^*, g^{x^*} \cdot g^s) &= e(g, g) \\ e(W_1^*, g^{r_j \cdot \mathsf{Ch}_{\mathcal{X}_j}(s)})e(W_2^*, g^{(x^*+s)}) &= e(g, g) \\ e(W_1^*, g^{r_j(x^*+s)q(s)})e(W_2^*, g^{(x^*+s)}) &= e(g, g) \\ [e(W_1^*, g^{r_j \cdot q(s)})e(W_2^*, g)]^{(x^*+s)} &= e(g, g) \\ [e(W_1^*, g^{r_j \cdot q(s)})e(W_2^*, g)] &= e(g, g)^{1/(x^*+s)} \end{aligned}$$

Observe that in both cases the left hand of the above equations is efficiently computable with access to $pub, (g^s, \ldots, g^{s^N}), r_j, \mathcal{X}_j, x^*, \mathsf{w}^*$. Hence, whenever Adv' succeeds in breaking the soundness of our scheme, Adv' outputs a pair breaking the q-SBDH assumption for $q = N$. By assumption the latter can happen only with negligible probability, and our claim that our scheme has soundness follows by contradiction. ∎

Proof of Zero-Knowledge. We define simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ as follows. At all times, we assume state_S contains all variables seen by the simulator this far.

- Sim_1 runs $\mathsf{GenParams}$ to receive pub. He then picks $s \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and sends g, g^s, pub to Adv. After Adv has output his set choice $\mathcal{X}$, Sim_1 picks $r \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and responds with $\mathsf{acc} = g^r$. Finally, he stores r and initiates empty list $\mathcal{C}$.
- For $i = 1, \ldots, l$ upon input (op, x_i, c_i):
  - If $\mathsf{op} = \mathsf{query}$, the simulator checks if $x_i \in \mathcal{C}$.
    - ∗ If $x_i \notin \mathcal{C}$, then if $D(\mathsf{query}, x_i, \mathcal{X}) = 1$, he computes $\kappa = r \cdot (x_i + s)^{-1}$ and responds with $(b = 1, \mathsf{w} = g^{\kappa})$. Else, if $D(\mathsf{query}, x_i, \mathcal{X}) = 0$ he computes q_1, q_2 such that $q_1 \cdot r + q_2 \cdot (x_i + s) = 1$, picks $\gamma \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and responds with $(b = 0, \mathsf{w} = (W_1 = g^{q_1 + \gamma(x_i+s)}, W_2 = g^{q_2 - \gamma r}))$. In both cases, the simulator appends (x_i, b, w) to $\mathcal{C}$.
    - ∗ If $x_i \in \mathcal{C}$ he responds with the corresponding entries b, w from $\mathcal{C}$.

- If $\mathsf{op} = \mathsf{update}$ then the simulator proceeds as follows. If $D(\mathsf{update}, x_i, c_i, \mathcal{X}) = 0$ then he responds with $\perp$. Else, he picks $r' \xleftarrow{\$} \mathbb{Z}_p^*$ and responds with $\mathsf{acc} = g^{r'}$. Finally he sets $r \leftarrow r'$ and $\mathcal{C} \leftarrow \emptyset$.

The simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ produces a view that is identically distributed to that produced by the challenger during $\mathsf{Real}_{\mathsf{Adv}}$. Observe that random values r are chosen independently after each update (and initial setup) in both cases. Once s, r are fixed then for any possible choice of $\mathcal{X}$ there exists unique $r^* \in \mathbb{Z}_p^*$ such that $g^r = g^{r^* \cdot \mathsf{Ch}_{\mathcal{X}}(s)}$. It follows that the accumulation values in $\mathsf{Real}_{\mathsf{Adv}}$ are indistinguishable from the (truly random) ones produced by Sim. For fixed s, r, given a set-element combination $(\mathcal{X}, x_i)$ with $x_i \in \mathcal{X}$, in each game there exists a unique membership witness w that satisfies verification. For negative witness $\mathsf{w} = (W_1, W_2)$, given a set-element combination $(\mathcal{X}, x_i)$ with $x_i \notin \mathcal{X}$, for each possible independently chosen value of γ, in both games there exists only one distinct corresponding pair W_1, W_2 that satisfies the verifying equation. It follows that the distributions in Definition 4 are identical and our scheme is perfect zero-knowledge. ∎

Efficiency comparison with the bilinear accumulator of [53]**.** Here we compare the efficiency of our accumulator with the bilinear accumulator of [53] –as extended in [2]– which is secure under the same assumption, but does not offer privacy. In Fig. 3, we show the number of necessary cryptographic operations for the constructions. We denote by $\mathsf{ADD}, \mathsf{MUL}$ point addition and scalar multiplication in the elliptic curve group $\mathbb{G}$, by ADD_T point addition in $\mathbb{G}_T$ and by PAIR a bilinear pairing computation. We stress that we do not measure the number of "non-cryptographic" operations, i.e., additions and multiplications modulo p.

As can be seen, our construction requires the same number of cryptographic operations for setup and membership witness construction and verification.

	[53]	**This paper**
Setup	$n\mathsf{MUL}$	$n\mathsf{MUL}$
Update	$1\mathsf{MUL}$	$2\mathsf{MUL}$
Witness (Member)	$n\mathsf{MUL} + (n-1)\mathsf{ADD}$	$n\mathsf{MUL} + (n-1)\mathsf{ADD}$
Witness (Non-Member)	$n\mathsf{MUL} + (n-1)\mathsf{ADD}$	$(n+1)\mathsf{MUL} + (n-1)\mathsf{ADD}$
Verify (Member)	$1(\mathsf{MUL} + \mathsf{ADD} + \mathsf{PAIR})$	$1(\mathsf{MUL} + \mathsf{ADD} + \mathsf{PAIR})$
Verify (Non-Member)	$2(\mathsf{MUL} + \mathsf{ADD} + \mathsf{PAIR})$	$1(\mathsf{MUL} + \mathsf{ADD} + \mathsf{ADD}_T) + 2\mathsf{PAIR}$
Witness Update (Member)	$1(\mathsf{MUL} + \mathsf{ADD})$	$2\mathsf{MUL} + 1\mathsf{ADD}$
Witness Update (Non-Member)	$2\mathsf{MUL} + 1\mathsf{ADD}$	$(n+1)\mathsf{MUL} + (n-1)\mathsf{ADD}$

Fig. 3. This table compares the number of cryptographic operations involved in each operation between our construction and that of [53] as extended in [2]. $\mathsf{ADD}, \mathsf{MUL}$ denote point addition and scalar multiplication in the elliptic curve group $\mathbb{G}$, ADD_T point addition in $\mathbb{G}_T$ and PAIR a pairing computation, whereas n is the size of the set.

For all other algorithms, the additional number of operations is only a constant (at most one) highlighting that zero-knowledge is achieved in practice with only a very small overhead[11]. The only notable exception is the update of non-membership witnesses in which case our construction resorts to re-computation from scratch.

Proving (non-)membership in batch. Another important property of our construction is that it allows the server to efficiently prove statements in batch. Consider the case when a client wants to issue a query on every element of a set $(y_1, \ldots, y_m)$. One way to achieve this would be to provide a separate membership/non-membership witness. This approach would yield a proof that consists of $O(m)$ group elements. Instead, with our construction the server can produce a single membership witness for all $y_i \in \mathcal{X}$ and a single non-membership witness for those $\notin \mathcal{X}$. We will use this technique for our construction in Sect. 5.

5 Zero-Knowledge Authenticated Set Collection (ZKASC)

Zero-knowledge accumulators, as presented so far, can be viewed as zero-knowledge authenticated sets where authenticated zero-knowledge membership/non-membership queries are supported on an outsourced set. In this section, we generalize the problem of zero-knowledge authentication from a set to a collection of sets to support outsourced set algebra operations: *is-subset, intersection, union and set difference.* We refer to this primitive as *zero-knowledge authenticated set collection* (ZKASC) since it falls in the general model of zero-knowledge authenticated data structures [29].

We consider a *dynamic collection* $\mathbb{S}$ of m sets $\mathcal{X}_1, \ldots, \mathcal{X}_m$, with elements from $\mathbb{X}$, that is remotely stored with an untrusted server. $\mathbb{S}$ has two types of operations defined on it: immutable operations $Q()$ and mutable operations $U()$. $Q(\mathbb{S}, q)$ takes a set algebra query q (wrt the indices of $\mathbb{S}$) as input and returns an answer and a proof and it does not alter $\mathbb{S}$. $U(\mathbb{S}, u)$ takes as input an update request and changes $\mathbb{S}$ accordingly. An update $u = (x, \mathsf{upd}, i)$ is either an insertion (if $\mathsf{upd} = 1$) of an element x into a set $\mathcal{X}_i$ or a deletion (if $\mathsf{upd} = 0$) of x from $\mathcal{X}_i$.

ZKASC is a tuple of six probabilistic polynomial time algorithms $\mathsf{ZKASC} = (\mathsf{KeyGen}, \mathsf{Setup}, \mathsf{Update}, \mathsf{UpdateServer}, \mathsf{Query}, \mathsf{Verify})$. Informally, ZKASC lets the owner outsource $\mathbb{S}$ and some auxiliary information and an evaluation key ek to the server (using $\mathsf{KeyGen}, \mathsf{Setup}$) and publish a verification key vk and public digest for $\mathbb{S}$. Then, the client can query $\mathbb{S}$ by sending queries to the server. For each query, the server generates answer and prepares its proof (using Query). The owner can also update his set collection and make corresponding changes

[11] Note however, that computing the coefficients of the polynomials that will be encoded in the exponents of the witnesses requires different types of polynomial arithmetic operations. In our construction the server runs an Extended Euclidean algorithm on input two polynomials of degree n and 1 respectively whereas in [2] he runs a polynomial division on the same inputs.

to digest (using Update) and the changes are propagated by the server to his copy of $\mathbb{S}$ and auxiliary information and ek (using UpdateServer). The client verifies the query answer against proof and the digest corresponding to the latest update using vk (in Verify). The security properties of ZKASC are: completeness, soundness and zero-knowledge. They are similar to those of ZKUA as described in Sect. 3, since both follow definition of ZKADS [29].

In the rest of the section we informally introduce our efficient construction of ZKASC, present the main theorem and compare the asymptotic complexity of the algorithms of our ZKASC scheme with that of [56] in Fig. 4. Our construction makes use of *zero-knowledge dynamic universal accumulator* introduced in Sect. 3 and *accumulation tree* described in Sect. 2. For the detailed algorithms and their security analysis we refer the reader to the full version.

5.1 Setup and Update Algorithms

The construction uses $pub = (p, \mathbb{G}, \mathbb{G}_T, e, g)$ as in Sect. 4. The owner runs Setup algorithm with the secret key s, the verification key (g^s, pub) (after generating them using KeyGen) and the set collection $\mathbb{S}$ as input and generates a short public digest for the client, the evaluation key ek and some authentication information of $\mathbb{S}$ for the server. The algorithm computes $\mathsf{acc}(\mathcal{X}_i)$ (zero-knowledge accumulation using Setup algorithm of Sect. 4) for each set $\mathcal{X}_i \in \mathbb{S}$. It then builds an accumulation tree on $\mathsf{acc}(\mathcal{X}_1) \ldots \mathsf{acc}(\mathcal{X}_m)$ and publishes the root of this tree as the public digest of $\mathbb{S}$. It sets the evaluation key to $(g, g^s, \ldots, g^{s^N})$ where $N = \sum_{i \in [1,m]} |\mathcal{X}_i|$. The auxiliary information for the server contains the randomness used for computing each $\mathsf{acc}(\mathcal{X}_i)$.

Update algorithm takes as input an update string u and updates the corresponding set in the set collection (using the Update algorithm in Sect. 4) and accordingly updates the authentication path in the accumulation tree and the auxiliary information, (possibly) the evaluation key and the public digest. As described so far, the update does not guarantee zero-knowledge. If a client queries wrt some set $j \neq i$ before and after u was performed, and sees that $\mathsf{acc}(\mathcal{X}_j)$ has not changed, then he learns that $\mathcal{X}_j$ is not affected by the update. This will also imply that the proofs that the client holds wrt $\mathcal{X}_j$ between updates are still valid. To achieve zero-knowledge, we require Update to re-randomize all the accumulation values that the client has seen (due to queries) since the last update. The update involves changes to authentication information stored with the server. To this end, the server runs UpdateServer algorithm to propagate owner's update on the set collection and authentication information. This algorithm updates the relevant set and updates all the authentication paths in the accumulation tree corresponding to the sets whose accumulation value has been changed or refreshed by the owner.

5.2 Set Algebra Query and Verify Algorithms

Query and Verify algorithms let the server construct a proof of a response to a set operation query and the client verify it, respectively. Since ZKASC supports several set operations, we describe each algorithm in terms of modular subroutines.

Is-subset query: A subset query $q = (\Delta, i)$ is parametrized by a set of elements Δ and an index i of a set collection. Given q, the subset query returns $\mathsf{answer} = 1$ if $\Delta \subseteq \mathcal{X}_i$ and $\mathsf{answer} = 0$ if $\Delta \nsubseteq \mathcal{X}_i$. This query is an efficient generalization of Witness (Sect. 4) where membership/non-membership query is supported for a batch of elements instead of a single element. The proof technique is similar to the membership and non-membership proof generation for a single element using Witness algorithm.

Set intersection query: Set intersection query $q = (i_1, \ldots, i_k)$ is parameterized by a set of indices of the set collection. The answer to an intersection query is a set of elements which we denote as answer and a simulatable proof of the correctness of the answer. If the intersection is computed correctly then $\mathsf{answer} = \mathcal{X}_{i_1} \cap \mathcal{X}_{i_2} \cap \ldots \cap \mathcal{X}_{i_k}$. We express the correctness of intersection with the two following conditions as in [56]:

Subset condition: $\mathsf{answer} \subseteq \mathcal{X}_{i_1} \wedge \ldots \wedge \mathsf{answer} \subseteq \mathcal{X}_{i_k}$. This condition ensures that the returned answer is a subset of all the queried set indices, i.e., every element of answer belongs to each set in the query.

Completeness condition: $(\mathcal{X}_{i_1} - \mathsf{answer}) \cap \ldots \cap (\mathcal{X}_{i_k} - \mathsf{answer}) = \emptyset$. This ensures that answer indeed contains *all* the common elements of $\mathcal{X}_{i_1}, \ldots, \mathcal{X}_{i_k}$, i.e., none of the elements have been omitted from answer.

To prove the first condition, we will use subset query as a subroutine. Proving the second condition is more tricky; it relies on the fact that the characteristic polynomials for the sets $\mathcal{X}_j - \mathsf{answer}$, for all $j \in [i_1, i_k]$, do not have common factors. In other words, these polynomials should be co-prime and their GCD should be 1 (Lemma 1). Since the proof units should be simulatable, we cannot directly use the technique as in [56]. To this end, we randomize the proof units by generalizing the randomization technique in Sect. 4 used to prove non-membership in a single set. The technique essentially adds noise in the exponent for each unit of the intersection proof such that they cancel out when used by the client in the bilinear map equality check.

Set union query: Set union query $q = (i_1, \ldots i_k)$ is parameterized by a set of indices of the set collection. The answer to a union query contains a set of elements, denoted as $\mathsf{answer} = \mathcal{X}_{i_1} \cup \mathcal{X}_{i_2} \cup \ldots \cup \mathcal{X}_{i_k}$, and a simulatable proof of the correctness of the answer. We *introduce* a technique for checking correctness of union operation based on the following conditions:

Superset condition: $\mathcal{X}_{i_1} \subseteq \mathsf{answer} \wedge \mathcal{X}_{i_2} \subseteq \mathsf{answer} \wedge \ldots \wedge \mathcal{X}_{i_k} \subseteq \mathsf{answer}$. This condition ensures that no element has been excluded from the returned answer.

Membership condition: $\mathsf{answer} \subseteq \tilde{U}$ where $\tilde{U} = \mathcal{X}_{i_1} \uplus \mathcal{X}_{i_2} \uplus \ldots \uplus \mathcal{X}_{i_k}$. $\uplus$ denotes multiset union of the sets, i.e., $\uplus$ preserves the multiplicity of every element in the union. This condition ensures that every element of answer belongs to *at least* one of the sets $\mathcal{X}_{i_1}, \ldots, \mathcal{X}_{i_k}$. We note that the trivial way (used in [56]) of proving this condition is to prove that each element of answer is a member of $\mathcal{X}_j$ for $j \in [i_1, i_k]$. This technique obviously does not support zero knowledge as it reveals which set the element comes from.

The first condition can be checked by using the subset proof as a subroutine.[12] The second condition should be proved carefully and not reveal (1) whether an element belongs to more than one of the sets in the query, and (2) which set an element in the union comes from. For example, returning $\tilde{U}$ in the clear trivially reveals the multiplicity of every element in answer. Instead, we request the server to return $\mathsf{acc}(\tilde{U})$ which equals $g^{\mathsf{Ch}_{\tilde{U}}(s)}$ blinded with randomness in the exponent. In order to prove that the server computed $\mathsf{acc}(\tilde{U})$ correctly, we introduce a *union tree.*

A union tree (UT) is a binary tree computed as follows. Corresponding to the k queried indices, $\mathsf{acc}(\mathcal{X}_{i_1}), \ldots, \mathsf{acc}(\mathcal{X}_{i_k})$ are the leaves of UT. The leaves are computed bottom up. Every internal node v is computed as follows. Let v_1 and v_2 be its two children. The (multi)set associated with v is the multiset $M = M_1 \uplus M_2$ where M_1 and M_2 are (multi)sets for v_1 and v_2 respectively. Let r_1 and r_2 be the blinding factors used in computing the accumulation values of v_1 and v_2, respectively. Then the node v stores value $a(v) = g^{r_1 r_2 \mathsf{Ch}_M(s)}$. Finally, the server constructs a proof of subset for answer in $\tilde{U}$.

The client can verify the correctness of each node of UT bottom up using a bilinear map as follows: $e(a(v), g) \stackrel{?}{=} e(a(v_1), a(v_2))$, where g is a part of the verification key. The membership proof verification of $\mathcal{X}_j \subseteq \mathsf{answer}$, $\forall j \in [i_1, i_k]$, and $\mathsf{answer} \subseteq \tilde{U}$ is done using subset verification subroutine.

Set difference query: Set difference query q is parameterized by two set indices of the set collection $q = (i_1, i_2)$. The answer to a set difference query is $\mathsf{answer} = \mathcal{X}_{i_1} - \mathcal{X}_{i_2}$ and a proof of correctness of the answer. We express the correctness of the answer using the following statement: $(\mathsf{answer} = \mathcal{X}_{i_1} - \mathcal{X}_{i_2}) \iff \mathcal{X}_{i_1} \setminus answer = \mathcal{X}_{i_1} \cap \mathcal{X}_{i_2}$. It ensures two conditions: (1) all the elements of answer indeed belong to $\mathcal{X}_{i_1}$ and (2) *all* the elements of $\mathcal{X}_{i_1}$ that are not in $\mathcal{X}_{i_2}$ are contained in answer. In other words, the union of answer and the intersection $I = \mathcal{X}_{i_1} \cap \mathcal{X}_{i_2}$ equals $\mathcal{X}_{i_1}$.

The second condition is tricky to prove for the following reasons. The server can reveal neither $\mathcal{X}_{i_1} - \mathsf{answer}$ nor $\mathcal{X}_{i_1} \cap \mathcal{X}_{i_2}$ to the client, since this reveals more than the set difference answer the client requested for (hence, breaking our zero-knowledge property).[13] Hence, we are required to provide blinded accumulators corresponding to these sets. Unfortunately, the blinded version of $\mathcal{X}_{i_1} \setminus answer =$

[12] We note that even the security proof does not assume the security proof for subset in a blackbox fashion since here it is the superset rather than the subset that is the known answer.

[13] We note that the sets are revealed to the client in [56] where privacy is not a concern.

$\mathcal{X}_{i_1} \cap \mathcal{X}_{i_2}$, even if the server computed them correctly, would be different. This is caused by different blinding factors used for these accumulators, even though the exponent that corresponds to the elements of the sets is the same. We use the latter fact and require the server to prove that the non-blinded exponents are the same. For this we use standard Schnorr proofs that can be made NIZKPoK in the Common Reference String model using standard techniques [22,28,46]. We describe the properties of a particular NIZKPoK protocol for discrete log in the full version [30]. We can now state the following result. The security proof and an efficiency analysis can be found in [30].

Theorem 3. *The scheme* ZKASC = (KeyGen, Setup, Update, UpdateServer, Query, Verify) *has perfect completeness, soundness under the q-SBDH assumption (with q set to the sum of maximum set sizes produced during the soundness game) and perfect zero-knowledge. Let $\mathbb{S} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ be the original set collection. Define $M = \sum_{i \in m} |\mathcal{X}_i|$, $n_j = |\mathcal{X}_j|$, and $N = \sum_{j \in [i_1, i_k]} n_j$. Let k be the number of group elements in the query input (for the subset query, it is the cardinality of a queried subset, and for the rest of the queries it is the number of set indices). Let ρ be the size of a query answer, L be the number of sets touched by the queries between updates u_{t-1} and u_t, and $0 < \epsilon < 1$ be a constant chosen at the time of setup. We have:*

- KeyGen *has complexity $O(1)$;*
- Setup *has complexity $O(M + m)$;*
- Update *and* UpdateServer *have complexity $O(L)$;*
- Query *and* Verify *have the following complexity:*
  - *For is-subset, the complexity is $O(N \log^2 N \log \log N + m^\epsilon \log m)$. The proof size is $O(k)$ and the verification has complexity $O(k)$.*[14]
  - *For set intersection, the complexity is $O(N \log^2 N \log \log N + km^\epsilon \log m)$. The proof size is $O(\rho + k)$ and the verification has complexity $O(\rho + k)$.*
  - *For set union, the complexity is $O(k\rho \log \rho + N \log N \log k + km^\epsilon \log m)$. The proof size is $O(\rho + k)$ and the verification has complexity $O(\rho + k)$.*
  - *For set difference, the complexity is $O(N \log^2 N \log \log N + m^\epsilon \log m)$. The proof size is $O(\rho)$ and the verification has complexity $O(\rho)$.*

5.3 Efficiency Comparison with the Scheme of [56]

We compare the asymptotic complexity of the algorithms of our ZKASC scheme with that of [56] in Fig. 4, which provides *only authenticity* and *trivially reveals information about the set collection*. We show that *only* update algorithms are more expensive compared to that of [56]. The extra cost is to achieve zero-knowledge, which requires all the proofs be ephemeral, i.e., proofs should not hold good between updates. We defer a more detailed comparison to the full version.

[14] Note that if the subset query is of the form: is set at index i a subset of the set at index j, then the proof complexity can be made constant.

		[56] \ This paper
Setup		$M+m$
Update	Owner	$1 \setminus L$
	Server	$1 \setminus L$
Subset	Query	$N\log^2 N \log\log N + m^{\varepsilon}\log m$
	Verify/Proof size	k
Instersection	Query	$N\log^2 N \log\log N + km^{\varepsilon}\log m$
	Verify/Proof size	$\rho + k$
Union	Query	$kN\log N + km^{\varepsilon}\log m$
	Verify/Proof size	$\rho + k$
Difference	Query	$N\log^2 N \log\log N + m^{\varepsilon}\log m$
	Verify/Proof size	ρ

Fig. 4. This table compares the asymptotic complexity of each operation with that of [56]. When only one value appears in the last column, it applies to both constructions. We note that the complexity of Union Query was originally mistakenly reported as $O(N \log N)$ in [56]. Notation: $m = |\mathbb{S}|$, $M = \sum_{i \in m} |\mathcal{X}_i|$, $n_j = |\mathcal{X}_j|$, $N = \sum_{j \in [i_1, i_k]} n_j$, k is the number of group elements in the query input (for the subset query it is the size of a queried subset Δ and for the rest of the queries it is the number of set indices), ρ is the size of the answer, L is the number of sets touched by queries between updates u_{t-1} and u_t, and $0 < \epsilon < 1$ is a constant chosen during setup.

6 Conclusion

In this work, we introduced the property of zero-knowledge for cryptographic accumulators. This is a strong privacy property, requiring that witnesses and accumulation values leak nothing about the accumulated set at any given point in the protocol execution, even after insertions and deletions. We showed that zero-knowledge accumulators are located between zero-knowledge sets and the recently introduced notion of primary-secondary-resolver membership proof systems, as the they can be constructed (in a black-box manner) from the former and they can be used to construct (in a black-box manner) the latter. We also presented a construction of an accumulator that achieves computational soundness and perfect zero-knowledge. Using this construction as a building block, we have designed a zero-knowledge authenticated set collection scheme that handles set-related queries that go beyond set (non-)membership. In particular, our scheme supports set unions, intersections, and differences, thus offering a complete set algebra. Future directions in the area include developing constructions that support efficient witness update, constructions based on constant-size assumptions (such as RSA) and constructing an efficient non-interactive set-difference protocol that does not rely on NIZKPoK's. Another interesting future direction is to equip zero-knowledge accumulators with zero-knowledge proofs of knowledge for membership/non-membership.

Acknowledgments. We thank the reviewers for their insightful comments and suggestions. We also thank Markulf Kohlweiss, Leonid Reyzin and Asaf Ziv for helpful discussions. This research was supported in part by the U.S. National Science Foundation under CNS grants 1012798, 1012910, 1228485 and 1645661.

References

1. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000). doi:10.1007/3-540-44598-6_16
2. Au, M.H., Tsang, P.P., Susilo, W., Mu, Y.: Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 295–308. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00862-7_20
3. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_33
4. Benaloh, J., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
5. Blanton, M., Aguiar, E.: Private and oblivious set and multiset operations. In: ASIACCS (2012)
6. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_4
7. Buldas, A., Laud, P., Lipmaa, H.: Accountable certificate management using undeniable attestations. In: CCS (2000)
8. Camacho, P., Hevia, A.: On the impossibility of batch update for cryptographic accumulators. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 178–188. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14712-8_11
9. Camacho, P., Hevia, A., Kiwi, M., Opazo, R.: Strong accumulators from collision-resistant hashing. In: Information Security (2008)
10. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00468-1_27
11. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002). doi:10.1007/3-540-45708-9_5
12. Canetti, R., Paneth, O., Papadopoulos, D., Triandopoulos, N.: Verifiable set operations over outsourced databases. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 113–130. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_7
13. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36362-7_5
14. Catalano, D., Fiore, D., Messina, M.: Zero-knowledge sets with short proofs. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 433–450. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_25

15. Chase, M., Healy, A., Lysyanskaya, A., Malkin, T., Reyzin, L.: Mercurial commitments with applications to zero-knowledge sets. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 422–439. Springer, Heidelberg (2005). doi:10.1007/11426639_25
16. Chatterjee, S., Menezes, A.: On cryptographic protocols employing asymmetric pairings - the role of ψ revisited. Discrete Appl. Math. **159**(13), 1311–1322 (2011)
17. Cristofaro, E., Tsudik, G.: Practical private set intersection protocols with linear complexity. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 143–159. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14577-3_13
18. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient robust private set intersection. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 125–142. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01957-9_8
19. Damgård, I., Triandopoulos, N.: Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538 (2008)
20. de Meer, H., Liedel, M., Pöhls, H.C., Posegga, J.: Indistinguishability of one-way accumulators. In Technical Report MIP-1210, Faculty of Computer Science and Mathematics (FIM), University of Passau (2012)
21. de Meer, H., Pöhls, H.C., Posegga, J., Samelin, K.: Redactable signature schemes for trees with signer-controlled non-leaf-redactions. In: E-Business and Telecommunications (2014)
22. Santis, A., Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_33
23. Derler, D., Hanser, C., Slamanig, D.: Revisiting cryptographic accumulators, additional properties and relations to other primitives. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 127–144. Springer, Heidelberg (2015). doi:10.1007/978-3-319-16715-2_7
24. Dong, C., Chen, L., Wen, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: ACM CCS (2013)
25. Fauzi, P., Lipmaa, H., Zhang, B.: Efficient non-interactive zero knowledge arguments for set operations. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 216–233. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45472-5_14
26. Fazio, N., Nicolosi, A.: Cryptographic accumulators: Definitions, constructions and applications. In Technical report. Courant Institute of Mathematical Sciences, New York University (2002)
27. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_1
28. Garay, J.A., MacKenzie, P., Yang, K.: Strengthening zero-knowledge protocols using signatures. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 177–194. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_11
29. Ghosh, E., Goodrich, M.T., Ohrimenko, O., Tamassia, R.: Verifiable zero-knowledge order queries and updates for fully dynamic lists and trees. In: Zikas, V., Prisco, R. (eds.) SCN 2016. LNCS, vol. 9841, pp. 216–236. Springer, Heidelberg (2016). doi:10.1007/978-3-319-44618-9_12
30. Ghosh, E., Ohrimenko, O., Papadopoulos, D., Tamassia, R., Triandopoulos, N.: Zero-knowledge accumulators and set operations. ePrint, 2015/404 (2015)

31. Ghosh, E., Ohrimenko, O., Tamassia, R.: Efficient verifiable range and closest point queries in zero-knowledge. In: Privacy Enhancing Technologies Symposium (PETs) (2016)
32. Ghosh, E., Ohrimenko, O., Tamassia, R.: Zero-knowledge authenticated order queries and order statistics on a list. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 149–171. Springer, Heidelberg (2015). doi:10.1007/978-3-319-28166-7_8
33. Goldberg, S., Naor, M., Papadopoulos, D., Reyzin, L., Vasant, S., Ziv, A.: NSEC5: Provably preventing DNSSEC zone enumeration. Cryptology ePrint Archive, Report 2014/582 (2014)
34. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: STOC (1985)
35. Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 491–511. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45611-8_26
36. Hazay, C., Nissim, K.: Efficient set operations in the presence of malicious adversaries. J. Cryptology **25**(3), 383–433 (2012)
37. Huang, Y., Evans, D., Katz, J.: Private set intersection: are garbled circuits better than custom protocols? In: NDSS (2012)
38. Jarecki, S., Liu, X.: Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00457-5_34
39. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005). doi:10.1007/11535218_15
40. Kosba, A.E., Papadopoulos, D., Papamanthou, C., Sayed, M.F., Shi, E., Triandopoulos, N.: TRUESET: faster verifiable set computations. In: USENIX (2014)
41. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72738-5_17
42. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: from polynomial commitments to pairing-based accumulators from simple assumptions. In: ICALP (2016)
43. Libert, B., Yung, M.: Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 499–517. Springer, Heidelberg (2010). doi:10.1007/978-3-642-11799-2_30
44. Lipmaa, H.: Secure accumulators from euclidean rings without trusted setup. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 224–240. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31284-7_14
45. Liskov, M.: Updatable zero-knowledge databases. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 174–198. Springer, Heidelberg (2005). doi:10.1007/11593447_10
46. MacKenzie, P., Yang, K.: On simulation-sound trapdoor commitments. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 382–400. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_23
47. Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy (1980)
48. Micali, S., Rabin, M.O., Kilian, J.: Zero-knowledge sets. In: FOCS (2003)

49. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed e-cash from bitcoin. In: IEEE Symposium on Security and Privacy (2013)
50. Morselli, R., Bhattacharjee, S., Katz, J., Keleher, P.J.: Trust-preserving set operations. In: IEEE INFOCOM (2004)
51. Naor, M., Nissim, K.: Certificate revocation and certificate update. IEEE J. Sel. Areas Commun. **18**(4), 561–570 (2000)
52. Naor, M., Ziv, A.: Primary-secondary-resolver membership proof systems. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 199–228. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46497-7_8
53. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30574-3_19
54. Nyberg, K.: Commutativity in cryptography. In: 1st International Trier Conference in Functional Analysis (1996)
55. Nyberg, K.: Fast accumulated hashing. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 83–87. Springer, Heidelberg (1996). doi:10.1007/3-540-60865-6_45
56. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Optimal verification of operations on dynamic sets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 91–110. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_6
57. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Authenticated hash tables based on cryptographic accumulators. Algorithmica (2015)
58. Prabhakaran, M., Xue, R.: Statistically hiding sets. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 100–116. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00862-7_7
59. Preparata, F., Sarwate, D., I. U. A. U.-C. C. S. LAB: Computational Complexity of Fourier Transforms Over Finite Fields. DTIC, 1976
60. Reyzin, L., Yakoubov, S.: Efficient asynchronous accumulators for distributed PKI. In: Zikas, V., Prisco, R. (eds.) SCN 2016. LNCS, vol. 9841, pp. 292–309. Springer, Heidelberg (2016). doi:10.1007/978-3-319-44618-9_16
61. Samelin, K., Pöhls, H.C., Bilzhause, A., Posegga, J., Meer, H.: Redactable signatures for independent removal of structure and content. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 17–33. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29101-2_2
62. Sander, T.: Efficient accumulators without trapdoor. In: ICICS (1999)
63. Tamassia, R.: Authenticated data structures. In: Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 2–5. Springer, Heidelberg (2003). doi:10.1007/978-3-540-39658-1_2
64. Zheng, Q., Xu, S.: Verifiable delegated set intersection operations on outsourced encrypted data. IACR Cryptology ePrint Archive (2014)