Zero-Knowledge Functional Elementary Databases

Xinxuan Zhang$^{1,2[0000-0002-2739-7656]}$ and Yi Deng$^{1,2[0000-0001-5948-0780]}$

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{zhangxinxuan, deng}@iie.ac.cn

Abstract. Zero-knowledge elementary databases (ZK-EDBs) enable a prover to commit a database D of key-value (x, v) pairs and later provide a convincing answer to the query "send me the value $D(x)$ associated with x" without revealing any extra knowledge (including the size of D). After its introduction, several works extended it to allow more expressive queries, but the expressiveness achieved so far is still limited: only a relatively simple queries–range queries over the keys and values– can be handled by known constructions.

In this paper we introduce a new notion called *zero knowledge functional elementary databases* (ZK-FEDBs), which allows the most general functional queries. Roughly speaking, for any Boolean circuit f, ZK-FEDBs allows the ZK-EDB prover to provide convincing answers to the queries of the form "send me all records (x, v) in D satisfying $f(x, v) = 1$," without revealing any extra knowledge (including the size of D). We present a construction of ZK-FEDBs in the random oracle model and generic group model, whose proof size is only linear in the length of record and the size of query circuit, and is independent of the size of input database D.

Our technical constribution is two-fold. Firstly, we introduce a new variant of zero-knowledge sets (ZKS) which supports combined operations on sets, and present a concrete construction that is based on groups with unknown order. Secondly, we develop a tranformation that tranforms the query of Boolean circuit into a query of combined operations on related sets, which may be of independent interest.

1 Introduction

Zero-knowledge sets (ZKS) are a valuable primitive introduced by Micali *et al.* [MRK03], which enable a prover to commit a finite set S and later prove the membership or non-membership of any element without revealing any extra knowledge (including the size of the set). An Elementary Database (EDB) D is a partial function mapping a (sub)set of keys into values (i.e., a set of key-value pairs (x, v) such that no two pairs have equivalent keys but different values). As described in [MRK03], the concept of ZKS can be extended to the one called zero-knowledge elementary databases (ZK-EDBs), which allows the prover to commit an EDB D and later prove that "x belongs to the support of D and $D(x) = v$" or that "x does not belong to the support of D" without revealing any knowledge beyond that. A number of ZK-EDB constructions have since emerged such as updatable ZK-EDBs [Lis05], independent ZK-EDBs [GM06] and efficient ZK-EDBs[CFM08, LY10], but, most constructions follow the paradigm of Chase *et al.* [CHL$^+$05], which relies on a Merkle tree and mercurial commitment and is not suitable to support richer queries.

Libert *et al.* [LNTW19] recently introduced zero-knowledge expressive elementary databases (ZK-EEDBs) that support the following richer queries: a) range query $[a_x, b_x]$, to which prover responds with all records $(x, v) \in D$ whose key x lies within $[a_x, b_x]$; b) range query $[a_v, b_v]$, to which prover responds with all records $(x, v) \in D$ whose value v is within the range $[a_v, b_v]$, and c) natural combination of range query $[a_x, b_x] \times [a_v, b_v]$. These techniques can be further exploited to support several other interesting queries such as k-nearest neighbours and k-minimum/maximum.

Despite the advancements made thus far, the expressivity of the known ZK-EDBs constructions is still very limited. For example, known constructions cannot even handle the simple query "send me all records $(x, v) \in D$ where the last bit of value v is zero", let alone the general Boolean circuit f query that requests to return all records $(x, v) \in D$ satisfying $f(x, v) = 1$.

Besides the theoretical value in ZK-EDB, enabling general function queries will have many practical applications. For instance, append-only ZK-EDBs are recently used to construct Key Transparency (KT) systems [CDGM19, CDG$^+$22], which maintain an auditable directory of the pairs of user's ID and their public keys while securely answer the queries for public key associated with certain ID in a consistent manner, even when the service provider is untrusted. ZK-EDBs with more expresive queries can improve the functionality of the KT system, allowing clients to more flexibly query public keys. Specifically, users can add labels or other short information to their IDs, such as a CV of a job hunter. Clients can then send queries to the service provider to obtain all public keys associated with IDs that meet their requirements, e.g., an HR can query the service provider to get all job hunters' public keys whose CVs satisfy certain requirement.

1.1 Our contribution

In this paper, we introduce a new concept called zero-knowledge functional elementary databases (ZK-FEDBs), which allows the most general function queries. Specifically, ZK-FEDBs enable one to commit an elementary database D of key-value pairs $(x, v) \in \{0, 1\}^\ell \times \{0, 1\}^\ell$ and then, for any Boolean circuit $f : \{0, 1\}^{2\ell} \to \{0, 1\}$, convincingly answer the query "Send me all records $(x, v) \in D$ satisfying $f(x, v) = 1$", without revealing any extra knowledge (including the size of D).

We present a construction of ZK-FEDBs based on groups of unknown orders, and prove its security in the random oracle model and generic group model. Its proof size is only linear in the length of record and the size of circuit f, independent of the size of input database D. Prior to our approach, the most expressive queries achievable were limited to range queries over keys and values, as demonstrated by Libert *et al.*[LNTW19].

Our technical constribution is two-fold (explained in detail below). Firstly, we introduce a new variant of zero-knowledge sets (ZKS) which supports combined operations on sets, and present a concrete construction that is based on groups with unknown order. Secondly, we develop a tranformation that tranforms the query of Boolean circuit into a query of combined operations on related sets, which may be of independent interest.

1.2 Technique Overview

A naive attempt to construct ZK-FEDBs is to use zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs). Specifically, one can use a SNARK-friendly hash function alone or, like most exsisting ZK-EDBs, use a Merkle tree to create a commitment for the database, and then use zk-SNARKs to generate proofs for queries. However, almost all zk-SNARKs expose the length of the witness. And for the commitment methods mentioned above, the witness must include all records in database to ensure the correctness of function queries. Therefore, this attempt would fail due to the potential revelation of the database size. The same issue will also arise when using other general-purpose zero-knowledge protocols.

RSA accumulator as ZKS and its limitations. Our start point is RSA accumulator, which is close to ZKS except that it offers no privacy. Let $\mathcal{H}_{prime}$ be a hash function mapping an element in set S into a prime. RSA accumulator computes $\mathsf{g}^{\Pi_{i\in[m]}p_i}$ to commit set $S = \{x_i\}_{i\in[m]}$, where $p_i = \mathcal{H}_{prime}(x_i)$.

Now we consider the three basic set operations, i.e, intersection, union and set-difference, on accumulators. It is a widely used approach to reduce set operation relations to several simpler set relations [PTT11, GOP⁺16, ZKP17]. Taking intersection as an example. Note that, $I = S_0 \cap S_1$ if and only if there exists $J_0 := S_0 \backslash I$ and $J_1 := S_1 \backslash I$ such that (J_0, J_1) belongs to disjoint relation $\{(J_0, J_1) | J_0 \cap J_1 = \emptyset\}$, and both (S_0, I, J_0) and (S_1, I, J_1) belong to union among disjoint relation $\{(U, J_0, J_1) |\ U = J_0 \cup J_1 \ \wedge\ J_0 \cap J_1 = \emptyset\}$.

Thus, given three RSA accumulators $\mathsf{C}_I, \mathsf{C}_{S_0}, \mathsf{C}_{S_1}$ to sets I, S_0, S_1, proving $I = S_0 \cap S_1$ is equivalent to prove the following statements: there exist accumulators $\mathsf{C}_{J_0}, \mathsf{C}_{J_1}$ to sets J_0 and J_1 such that a) the *committed* sets (J_0, J_1) belongs to disjoint relation, and b) the *committed* sets (S_0, I, J_0) and (S_1, I, J_1) belong to union among disjoint relation. It is easy to verify that these two items a) and b) are equivalent to the following two conditions respectively:

$a')$ $(\mathsf{C}_{J_0}, \mathsf{C}_{J_1})$ belongs to co-prime relations $\{(\mathsf{C}_1, \mathsf{C}_2) | \exists a, b \in \mathbb{Z}\ s.t.\ \gcd(a, b) = 1 \wedge (\mathsf{C}_1, \mathsf{C}_2) = (\mathsf{g}^a, \mathsf{g}^b)\}$.

$b')$ Both $(\mathsf{C}_I, \mathsf{C}_{J_0})$ and $(\mathsf{C}_I, \mathsf{C}_{J_1})$ belong to co-prime relations, and both $(\mathsf{C}_{S_0}, \mathsf{C}_I, \mathsf{C}_{J_0})$ and $(\mathsf{C}_{S_0}, \mathsf{C}_I, \mathsf{C}_{J_1})$ are DDH tuples.

All of them can be proved easily relying on Boneh's PoKE (Proof of knowledge of exponent) protocol and its variants [BBF19]. The other two basic set-operation relations on accumulators, can also be proved in a similar manners.

As in [XLL07, XLL08], one can achieve privacy and obtain a ZKS scheme by using randomness r in computing the commitment $\mathsf{g}^{r\Pi_{i\in[m]}p_i}$ to S. However, the introduction of randomness would invalidate the proof of basic set-operation relations on commitments. Specifically, for this ZKS, the disjoint relations and the union among disjoint sets relations on *commited* sets can no longer be equivalent to co-prime relations and DDH relations over RSA groups.

Zero-Knowledge Sets with Set-Operation Queries. Our key observation is that the randomness r in above ZKS scheme can be chosen from small and bounded range of $[0, B]$. This leads to that, for sets S_0, S_1, U satisfying $S_0 \cap S_1 = \emptyset, S_0 \cup S_1 = U$ and their commitments $\mathsf{C}_{S_0}, \mathsf{C}_{S_1}, \mathsf{C}_U$, we have:

$a'')$ The greatest common divisor of the exponents of commitments $\mathsf{C}_{S_0}, \mathsf{C}_{S_1}$ is small. We call such a tuple $(\mathsf{C}_{S_0}, \mathsf{C}_{S_1})$ as a *pseudo-coprime exponent* tuple.

$b'')$ The commitment tuple $(\mathsf{C}_{S_0}, \mathsf{C}_{S_1}, \mathsf{C}_U)$ is close to a DDH tuple. We call such a tuple $(\mathsf{C}_{S_0}, \mathsf{C}_{S_1}, \mathsf{C}_U)$ as a *pseudo-DDH* tuple.

We present a series of NIZK protocols to prove the above pseudo-relations. Though these NIZK protocols achieve somewhat weaker soundness, they are sufficient for our applications.

We further consider more general combined operations on sets, and regard it as a "circuit" with gates "intersection", "union" and "set-difference" in a natural way. To construct a ZKS supporting combined operations on sets (i.e., a ZKS that allows prover to convincingly answer the query "send me all records in $\mathcal{Q}(S_1, \cdots, S_m)$" for any "circuit" $\mathcal{Q}$ and committed sets $\{S_i\}_{i\in[m]}$), the prover can use above NIZK proofs to demonstrate that each gate/set-operation is performed honestly.

From Boolean Circuit Queries to Set-Operation. A crucial step toward our construction of ZK-FEDBs is a transformation that transform a query of Boolean circuit f over a set S (requesting $S_{output} := \{x | x \in S \wedge f(x) = 1\}$) into a query of combined operations $\mathcal{Q}$ on related sets $S_i^b = \{x | x \in S \wedge \text{the i-th bit of "}x\text{" is } b\}$ (requesting $S_{output} := \mathcal{Q}(\{S_i^b\})$).

This transformation proceeds as follows. Let the number of input wires of f be n. We first associate each input wire i of f with two subsets $\{S_i^b\}_{b\in\{0,1\}}$ of S, which are definded as above. Sequentially, for each gate in f, we associate its output wire i $(i > n)$ with two subsets $\{S_i^b\}_{b\in\{0,1\}}$, which are defined in the following way:

- For an AND gate with two input wires a, b and an output wire c, the two sets associated with wire c are set to be $S_c^0 = S_a^0 \cup S_b^0$ and $S_c^1 = S_a^1 \cap S_b^1$.
- For an OR gate with two input wires a, b and an output wire c, the two sets associated with wire c are set to be $S_c^0 = S_a^0 \cap S_b^0$ and $S_c^1 = S_a^1 \cup S_b^1$.

- For a NOT gate with an input wire a and an output wire b, the two sets associated with wire c are set to be $S_b^0 = S_a^1$ and $S_b^1 = S_b^0$.

In the ending, the second set S_ℓ^1 associated with the output wire ℓ of f is now a result of a circuit $\mathcal{Q}$ of combined operations on the sets $\{S_i^b\}$ associated with the input wires, i.e., $S_\ell^1 = \mathcal{Q}(\{S_i^b\})$.

One can check that the above resulting set S_ℓ^1 is exactly the set $S_{output} := \{x | x \in S \wedge f(x) = 1\}$. A crucial observation here is that, for each x belonging to S and each wire i, $x \in S_i^b$ if and only if the value of i-th wire of $f(x)$ is b. Therefore S_ℓ^b is the set of x that makes the output wire of f equaling to b, which means that $S_\ell^1 = \{x | x \in S \wedge f(x) = 1\}$.

A simple example of transforming $f(x) = \bar{x}_1 \wedge \bar{x}_2 \vee (\neg \bar{x}_3)$ (where $x = \bar{x}_1 \| \bar{x}_2 \| \bar{x}_3 \in \{0,1\}^3$) is shown in Fig.1.

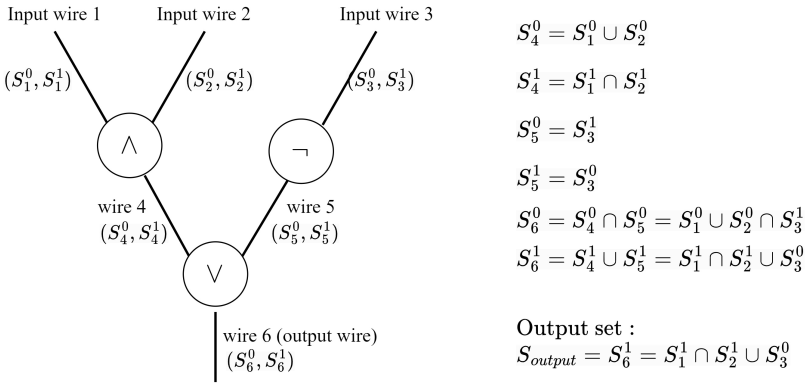

Fig. 1

Finally, we construct ZK-FEDBs using ZKS with set-operation queries and standard ZK-EDBs. Roughly, we use the former to ensure the correctness of function queries and the latter to ensure the correctness of associated values. Furthermore, we construct a constant-size ZK-EDB in conjunction with standard batch techniques, and achieve a ZK-FEDB with a proof size that is only linear in the length of the record and the size of the circuit f, and is independent of the size of the input database D.

1.3 Related Work

Since the notion of ZK-EDB was first introduced by Micali *et al.*[MRK03], numerous works concentrating on the performance, security, and functionality of ZK-EDB have been developed.

In [CHL$^+$05], Chase *et al.* introduced the notion of mercurial commitments and presented a widely used paradigm to construct a ZK-EDB. Mercurial commitments (and thus ZK-EDBs) can be constructed through one-way functions [CDV06], and efficient mercurial commitments (and thus efficient ZK-EDBs) can be constructed through DL, Factoring, RSA or LWE assumption [CDV06, Zhu09, LNTW19]. The notion of q-mercurial commitments were introduced and developed in [CFM08, LY10, CF13] to further compress the (non-)membership proof size of ZK-EDBs. Li *et al.* [LSY$^+$21] introduced concise mercurial subvector commitments and achieved batch verifiable ZK-EDBs.

There are also several works focusing on developing the security definition of ZK-EDBs, such as independent ZK-EDBs [GM06] and secure database commitments [CV12]. Prabhakaran and Xue [PX09] put forward statistically hiding set and present its construction from RSA accumulators. Following [PX09], [XLL07, XLL08] constructed a constant-size ZKS.

Another research point of ZK-EDBs is how to extend its functionality. Ostrovsky et al.[ORS04] explored generating consistency proofs for queries on a committed database and present a concrete constrction for range queries over the keys. Liskov [Lis05] presented updatable ZK-EDBs in the random oracle model. Ghosh *et al.* [GOT15] introduced zero-knowledge lists, which allow one to commit a list and later answer order queries in a convincing manner. Libert *et al.* [LNTW19] recently introduced ZK-EEDBs that support richer queries, e.g., range queries over the keys and values.

Accumulators (e.g. [BdM93, CL02, CHKO08, Ngu05, DHS15]) are an extremely well-studied cryptography primitive related to ZKS. Accumulators allow representing a set using an accumulation value and later providing (non-)membership proofs; however, *hiding and zero-knowledge properties are not necessary for accumulators.* Although Ghosh *et al.* [GOP$^+$16] and Zhang *et al.* [ZKP17] proposed the constructions of zero-knowledge accumulators supporting set operations, their schemes only consider collision-freeness security, where the adversary cannot cheat in a proof for an *honestly* generated accumulation value. In contrast, ZKS prevent the adversary from cheating in a proof even for a *maliciously* generated commitment. Agrawal and Raghuraman [AR20] proposed a commitment scheme for databases of key-value pairs. Their scheme does not provide privacy.

Authenticated data structures (ADS) (e.g., [Tam03, PTT11, NZ15]) also allow a trusted database owner to commit its database, and an untrusted server can answer the queries on behalf of trusted database owners to any clients knowing the commitment. However, as a three-party scheme in which the committer (database owner) is always trusted, ADS is incomparable to ZK-EDB.

Key Transparency (KT) systems (e.g. [MBB$^+$15, TBP$^+$19, HHK$^+$21, TKPS22]) allow service providers to maintain an auditable directory of their users' public keys, producing proofs that all participants have a consistent view of those keys, and allowing each user to check updates to their own keys. Recently, Chase *et al.*[CDGM19] show how to construct privacy-preserving KT systems from append only ZKS and Chen *et al.*[CDG+22] develop it and achieve post compromise security.

Recently, there are several works [GW20, ZBK+22, ZGK+22, EFG22] of lookup arguments studying how to efficiently prove subset relation and achieving constant proof size. Lookup arguments can be zero-knowledge and can be used alongside zk-SNARKs to prove that "all records in D_{output} satisfy the query function f and $D_{output} \subset D$" without revealing any extra knowledge. However, constructing ZK-FEDBs from lookup arguments and zk-SNARKs is considerably more challenging. The reason is that, to achieve ZK-FEDBs, one must also prove that any record in $D \backslash D_{output}$ does not satisfy the query function f. Utilizing zk-SNARKs in this context would leak the size of the remaining database $D \backslash D_{output}$ as we discussed in Technique Overview, and therefore leak the size of D.

1.4 Organization

Preliminaries are described in section 2. In section 3, we introduce several new building blocks. In section 4 we introduce and construct ZKS with set-operation queries. In section 5 we show how to transform a Boolean circuit and introduce the notion of ZK-FEDBs, while also providing a concrete construction. Due to space constraints, the construction of constant-size standard ZK-EDBs and several security proofs are deferred to Appendix.

2 Preliminaries

In this paper, we denote by λ the security parameter, by $[m]$ the set $\{1, 2, \cdots, m\}$ and by $[m_1, m_2]$ the set $\{m_1 + 1, m_1 + 2, \cdots, m_2\}$. A non-negative function $f : \mathbb{N} \to \mathbb{R}$ is negligible if $f(\lambda) = \lambda^{-w(1)}$. We use the standard abbreviation PPT to denote probabilistic polynomial time.

An elementary database D is a set of key-value pairs $(x, v) \in \{0, 1\}^\ell \times \{0, 1\}^\ell$ such that if $(x, v) \in D$ and $(x, v') \in D$, then $v = v'$. Here ℓ is a public polynomial in λ. We denote by $Sup(D)$ the support of D, i.e., the set of $x \in \{0, 1\}^\ell$ for which $\exists v$ such that $(x, v) \in D$. We denote such unique v as $D(x)$, and if $x \notin Sup(D)$, we then also write $D(x) =\perp$. For consistency, for any set S of elements $x \in \{0, 1\}^\ell$, we write $S(x) = 1$ if $x \in S$ and write $S(x) =\perp$ if $x \notin S$.

2.1 Zero-Knowledge Elementary Databases and Sets

ZKS allow one to commit a set S and later prove the (non-)membership of any elements without revealing any extra knowledge (including the size of the set). The notion of ZKS can be extended to ZK-EDBs, which allow one to commit an elementary database D. Due to that ZKS can be seen as a special case of ZK-EDBs, where $D(x) = 1$ if $x \in Sup(D)$, we skip the definition of ZKS here. Following [MRK03, GM06, LNTW19], we present the following formal definition of ZK-EDBs:

Definition 1 (Zero-Knowledge Elementary Database). *A zero-knowledge elementary database consists of four algorithms* (Setup, Com, Prove, Verify)*:*

- $\delta \leftarrow \mathsf{Setup}(1^\lambda)$*: On input the security parameter* 1^λ*,* Setup *outputs a random string (or a structured reference string)* δ *as the CRS.*
- $(com, \tau) \leftarrow \mathsf{Com}(\delta, D)$*: On input the CRS* δ *and an elementary database* D*,* Com *outputs a commitment of database com and an opening information* τ*.*
- $\pi \leftarrow \mathsf{Prove}(\delta, com, \tau, x, v)$*: On input the CRS* δ*, the pairing of the commitment and opening information* (com, τ)*, and a key* x *and its associated value* v *(i.e.,* $(x, v) \in D$ *or* $x \notin Sup(D), v = \perp$*),* Prove *outputs a proof* π *of* $v = D(x)$
- $0/1 \leftarrow \mathsf{Verify}(\delta, com, x, v, \pi)$*: On input the CRS* δ*, commitment com, key-value pair* (x, v) *and proof* π*,* Verify *either outputs* 1 *(denoting accept) or* 0 *(denoting reject).*

It satisfies the following three properties:

- **Completeness:** *For any elementary database* D *and any* x*,*

$$\Pr\left[\mathsf{Verify}(\delta, com, x, D(x), \pi) = 1 \middle| \begin{array}{c} \delta \leftarrow \mathsf{Setup}(1^\lambda); (com, \tau) \leftarrow \mathsf{Com}(\delta, D); \\ \pi \leftarrow \mathsf{Prove}(\delta, com, \tau, x, D(x)) \end{array}\right] = 1$$

- **Soundness:** *For any PPT adversary* $\mathcal{A}$*, there exists a negligible function* $negl(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{c} v \neq v' \wedge \\ \mathsf{Verify}(\delta, com, x, v, \pi) = 1 \wedge \\ \mathsf{Verify}(\delta, com, x, v', \pi') = 1 \end{array} \middle| \begin{array}{c} \delta \leftarrow \mathsf{Setup}(1^\lambda); \\ (com, x, v, v', \pi, \pi') \leftarrow \mathcal{A}(\delta) \end{array}\right] \leq negl(\lambda)$$

- **Zero-Knowledge:** *There exists a simulator Sim such that for any PPT adversary* $\mathcal{A}$*, the absolute value of the difference*

$$\Pr\left[\mathcal{A}^{\mathcal{O}_P}(\delta, state_{\mathcal{A}}, com) = 1 \middle| \begin{array}{c} \delta \leftarrow \mathsf{Setup}(1^\lambda), (D, state_{\mathcal{A}}) \leftarrow \mathcal{A}(\delta), \\ (com, \tau) \leftarrow \mathsf{Com}(\delta, D) \end{array}\right] -$$

$$\Pr\left[\mathcal{A}^{\mathcal{O}_S}(\delta, state_{\mathcal{A}}, com) = 1 \middle| \begin{array}{c} (\delta, state_\delta) \leftarrow Sim(1^\lambda), (D, state_{\mathcal{A}}) \leftarrow \mathcal{A}(\delta), \\ (com, state_S) \leftarrow Sim(\delta, state_\delta) \end{array}\right]$$

is negligible in λ*, where* $\mathcal{O}_P$ *and* $\mathcal{O}_S$ *are defined as follows:*
$\mathcal{O}_P$*: On input a string* x*,* $\mathcal{O}_P$ *outputs* $\pi \leftarrow \mathsf{Prove}(\delta, com, \tau, x, D(x))$*.*
$\mathcal{O}_S$*: On input a string* x*,* $\mathcal{O}_S$ *outputs* $\pi \leftarrow Sim(state_S, x, D(x))$*.*

2.2 Groups of Unknown-Order and Assumptions

In this paper, the schemes are constructed on groups of unknown order, for which the order is difficult to compute for the committer. Groups of unknown order are a useful tool in the construction of polynomial commitments, integer commitments, and accumulators, among other aspects.

The strong RSA assumption is a useful assumption for groups of unknown orders. We introduce it in the following.

Assumption 1. *(Strong RSA Assumption)[BP97, AR20]. The strong RSA assumption states that an efficient adversary cannot compute ℓ-th roots for a given random group element, where ℓ is an odd prime chosen by the adversary. Specifically, it holds for GGen if for any probabilistic polynomial time adversary $\mathcal{A}$,*

$$\Pr\left[\mathsf{u}^{\ell} = \mathsf{g} \textit{ and } \ell \textit{ is an odd prime} \;\middle|\; \begin{array}{l} \mathbb{G} \leftarrow GGen(\lambda), \mathsf{g} \xleftarrow{\$} \mathbb{G}, \\ (\mathsf{u}, \ell) \in \mathbb{G} \times \mathbb{N} \leftarrow \mathcal{A}(\mathbb{G}, \mathsf{g}) \end{array}\right] \leq negl(\lambda).$$

Generic group model. In this paper, we use the generic group model for groups of unknown order as defined by Damgård and Koprowski [DK02], and as used in [BBF19]. Portions of the definition of the generic group model are taken verbatim from [BBF19].

In the generic group model, the group is parameterized by two public integers A and B, and the group order is sampled uniformly from $[A, B]$. The group $\mathbb{G}$ is defined by a random injective function $\sigma : \mathbb{Z}_{|\mathbb{G}|} \rightarrow \{0,1\}^l$ for some l, where $2^l \gg |\mathbb{G}|$. The group elements are $\sigma(0), \cdots, \sigma(|\mathbb{G}|)$. A generic group algorithm $\mathcal{A}$ is a probabilistic algorithm. Let $\mathcal{L}$ be a list that is initialized with the encodings given to $\mathcal{A}$ as input. $\mathcal{A}$ can query two generic group oracles. The first oracle $\mathcal{O}_1$ samples a random $r \in \mathbb{Z}_{|\mathbb{G}|}$ and returns $\sigma(r)$, which is appended to the list of encodings $\mathcal{L}$. The second oracle $\mathcal{O}_2(i, j, \pm)$ takes two indices $i, j \in [p]$, where p is the size of $\mathcal{L}$, as well as a sign bit, and returns $\sigma(i \pm j)$, which is appended to $\mathcal{L}$. Note that herein $\mathcal{A}$ is not given the order of $\mathbb{G}$.

As shown in [DK02], the strong RSA assumption holds in the generic group model.

Zero-knowledge protocol for bounded discrete-log. The classical Schnorr Σ-protocol can be used to prove the discrete-log relation when the exponent is small (i.e., $\mathcal{R}_{boundedDL} = \{(\mathsf{u}, \mathsf{w}, T; x) | \mathsf{u}^x = \mathsf{w} \wedge |x| \leq T\}$). It only provides a weak soundness that, a proof for $(\mathsf{u}, \mathsf{w}, T)$ can convince verifier that $(\mathsf{u}, \mathsf{w}, T)$ belongs to a relaxed relation $\mathcal{R}^*_{boundedDL} = \{(\mathsf{u}, \mathsf{w}, T; x, t) | \mathsf{u}^x = \mathsf{w}^t \wedge |x| \leq 2^{2\lambda}T, |t| \leq 2^{\lambda}\}$, which is sufficient for our goal. Following [DF02, CS97, FO97], the construction is as follows.

Protocol $\mathsf{ZK}_{boundedDL}$ (Zero-knowledge protocol for $\mathcal{R}_{boundedDL}$)

Params: $\mathbb{G} \leftarrow GGen(\lambda)$; Common Input: $\mathsf{u}, \mathsf{w} \in \mathbb{G}, T \in \mathbb{N}$;
Private Input for $\mathcal{P}$: $x \in \mathbb{Z}$

1. $\mathcal{P}$ samples $r \xleftarrow{\$} [2^{2\lambda}T]$ and sends $\mathsf{z} = \mathsf{u}^r \in \mathbb{G}$ to $\mathcal{V}$.
2. $\mathcal{V}$ sends challenge $c \xleftarrow{\$} [2^{\lambda}]$.
3. $\mathcal{P}$ computes $s = r + cx$ and sends s to $\mathcal{V}$.
4. $\mathcal{V}$ accepts if $\mathsf{u}^s = \mathsf{z}\mathsf{w}^c$ and $s \leq 2^{2\lambda}T$.

Fig. 2: Protocol $\mathsf{ZK}_{boundedDL}$ [CS97, FO97]

Lemma 1. Protocol $\mathsf{ZK}_{boundedDL}$ *is an honest-verifier statistically zero-knowledge protocol for* $\mathcal{R}_{boundedDL}$*, achieving a weak knowledge soundness defined as follows: There exists an extractor such that for any polynomial* p *and any prover* $\mathcal{P}^*$ *convincing verifier of statement* $(\mathsf{u}, \mathsf{w}, T)$ *with probability* p^{-1}*, the extractor can extract* (x', t) *within an expected polynomial time such that* $|x'| \le 2^{2\lambda}T, |t| \le 2^{\lambda}$ *and* $\mathsf{u}^{x'} = \mathsf{w}^t$*.*

Note that the honest-verifier statistically zero-knowledge property of the above lemma directly follows [DF02]. And the weak knowledge soundness can be easily proved by rewinding.

Furthermore, above zero-knowledge protocol can be easily extend to multidimensional discrete-log relation with small exponents (i.e., $\mathcal{R} = \{(\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w}, T;$ $\{x_i\}_{i\in[n]})|\mathit{\Pi}_{i\in[n]}\mathsf{u}_i^{x_i} = \mathsf{w} \wedge \forall i \in [n], |x_i| \le T\}$), resulting in a similar weak knowledge soundness.

Proof of Knowledge of Exponent (PoKE). Recently, Boneh *et al.* [BBF19] introduced a way to present an argument of knowledge protocol for the following relation.

$$\mathcal{R}_{PoKE} := \{(\mathsf{u}, \mathsf{w} \in \mathbb{G}; x \in \mathbb{Z})|\mathsf{w} = \mathsf{u}^x \in \mathbb{Z}\}$$

Let $\mathcal{P}$ be the prover and $\mathcal{V}$ be the verifier. Let $\mathsf{Primes}(\lambda)$ denote the set of odd prime numbers in $[0, 2^{\lambda}]$. Their protocol is as follows:

Protocol PoKE (Proof of knowledge of exponent)

Params: $\mathbb{G} \leftarrow GGen(\lambda), \mathsf{g} \in \mathbb{G}$; Common Input: $\mathsf{u}, \mathsf{w} \in \mathbb{G}$;
Private Input for $\mathcal{P}$: $x \in \mathbb{Z}$

1. $\mathcal{P}$ sends $\mathsf{z} = \mathsf{g}^x \in \mathbb{G}$ to $\mathcal{V}$.
2. $\mathcal{V}$ samples $l \xleftarrow{\$} \mathsf{Primes}(\lambda)$ and sends l to $\mathcal{P}$.
3. $\mathcal{P}$ finds the quotient $q \in \mathbb{Z}$ and residue $r \in [l]$ such that $x = ql + r$. $\mathcal{P}$ sends $\mathsf{Q} = \mathsf{u}^q$, $\mathsf{Q}' = \mathsf{g}^q$, and r to $\mathcal{V}$.
4. $\mathcal{V}$ accepts if $r \in [l]$, $\mathsf{Q}^l\mathsf{u}^r = \mathsf{w}$, and $\mathsf{Q}'^l\mathsf{g}^r = \mathsf{z}$.

Fig. 3: PoKE protocol [BBF19]

Theorem 1 ([BBF19] Theorem 3.). Protocol PoKE *is an argument of knowledge for the relation* $\mathcal{R}_{PoKE}$ *in the generic group model.*

Our constructions will use above PoKE protocol as a subroutine, and that's why our results rely on the generic group model. In practice, there are two common methods used to instantiate groups of unknown order.

RSA group: The multiplicative group $\mathbb{Z}_n^*$ of integers modulo a product $n = pq$ of large primes p and q. Any efficient algorithm that calculates the order can be transformed into an efficient algorithm factoring n. In addition, we need to point out that it is difficult to generate the RSA group in a publicly verifiable way without exposing the order. Therefore, we need a trusted party to generate the group.

Class group: The class group of an imaginary quadratic order with discriminant Δ where $-\Delta$ is a prime and $\Delta \equiv 1 \mod 4$. As an important property, one can choose a security class group $Cl(\Delta)$ by choosing the "good" discriminant Δ randomly without a trusted party. For more details, one can refer to Buchmann and Hamdy's survey [BH01] and Straka's accessible blog post [Str19] for more details.

At the end of this section, we provide several simple lemmas used for our construction.

Lemma 2. *For any positive integers a, A, and B satisfying $B > A$, we have:*

$$\mathsf{Dist}(\{x \overset{\$}{\leftarrow} \mathbb{Z}_a\}, \{x \mod a | x \overset{\$}{\leftarrow} [A, B]\}) \leq \frac{a}{B - A}$$

where Dist *indicates the statistical distance between distributions.*

Lemma 3. *For any integers s_1,s_2 and positive integers a,A,B satisfying $B > A$,* $\gcd(s_1, s_2) = 1$*, we have:*

$$\mathsf{Dist}(\{x \overset{\$}{\leftarrow} \mathbb{Z}_a\}, \{xs_1 + ys_2 \mod a | x, y \overset{\$}{\leftarrow} [A, B]\}) \leq \frac{3a}{B - A}$$

where Dist *indicates the statistical distance between distributions.*

Lemma 4. *For any multiplicative group $\mathbb{G}$ and group elements $\mathsf{g}, \mathsf{h} \in \mathbb{G}$, if there exists* coprime *integers a, p satisfying $\mathsf{g}^a = \mathsf{h}^p$, then one can easily compute h' satisfying $\mathsf{g} = \mathsf{h}'^p$ from a, p, g and h.*

The proofs of the above three lemmas are shown in Appendix.A.

3 New Building Blocks

This section introduces several building blocks that we use in our construction. It comprises of two parts. In the first part, we present a new variant of Boneh *et al.*'s zero-knowledge protocol for multidimensional discrete-log relation and lightly modify the standard ZKS scheme [PX09, XLL07, XLL08]. In the second part, we construct two new zero-knowledge protocols for pseudo-coprime exponent relation and pseudo-DDH relation over the groups of unknown orders.

3.1 Zero-Knowledge Protocol for Multidimensional Discrete-log and Standard ZKS Scheme

In [BBF19], Boneh *et al.* combined the classical Schnorr Σ-protocol and the batched PoKE protocol to present a zero-knowledge argument of knowledge protocol (called Protocol ZKPoKRep) for the relation $\mathcal{R}_{multiDL} = \{(\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w}; \{x_i\}_{i\in[n]})$

$|\mathit{\Pi}_{i\in[n]}\mathsf{u}_i^{x_i} = \mathsf{w}\}$. Their protocol satisfies soundness only when $\{\mathsf{u}_i\}_{i\in[n]}$ is a base specified in the CRS. However, our constructions require the prover to generate such a set $\{\mathrm{u}_i\}_{i\in[n]}$.

Therefore, we construct a new variant of Boneh *et al.*'s protocol. We call it Protocol $\mathsf{ZK}_{multiDL}$. Compared to the origin protocol, the prover in our new protocol uses n PoKE protocols to prove the relation $\mathsf{zw}^c = \mathit{\Pi}_{i\in[n]}\mathsf{u}_i^{s_i}$. Here, we require the prover to send $\mathsf{u}_i^{s_i}$ additionally, which doesn't affect the proof of statistical honest verifier zero-knowledge. This allows the extractor to extract s_i even when $\{\mathsf{u}_i\}_{i\in[n]}$ is generated by the prover. As a result, we obtain a zero-knowledge protocol for $\mathcal{R}_{multiDL}$ where $\{\mathsf{u}_i\}_{i\in[n]}$ can be generated by the prover. However, this benefit comes at a price: This protocol only satisfies a weak knowledge soundness that a valid proof can convice the verifier that the statement belongs to a *relaxed* relation $\mathcal{R}^*_{multiDL} = \{(\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w}; \{x_i\}_{i\in[n]}, t)| \mathit{\Pi}_{i\in[n]}\mathsf{u}_i^{x_i} = \mathsf{w}^t, |t| \leq 2^\lambda\}$. The concrete construction is shown in Fig.4.

Protocol $\mathsf{ZK}_{multiDL}$ (Zero-knowledge protocol for $\mathcal{R}_{multiDL}$)

Params: $\mathbb{G} \leftarrow GGen(\lambda), B \geq |\mathbb{G}|$; Common Input: $\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w} \in \mathbb{G}$;
Private Input for $\mathcal{P}$: $\{x_i\}_{i\in[n]}$

1. $\mathcal{P}$ samples $r_i \xleftarrow{\$} [2^{2\lambda}B]$ for each $i \in [n]$ and sends $\mathsf{z} = \mathit{\Pi}_{i\in[n]}\mathsf{u}_i^{r_i} \in \mathbb{G}$ to $\mathcal{V}$.
2. $\mathcal{V}$ sends challenge $c \xleftarrow{\$} [2^\lambda]$.
3. $\mathcal{P}$ computes $s_i = r_i + cx_i$ for each $i \in [n]$ and sends $\hat{\mathsf{u}}_i = \mathsf{u}_i^{s_i}$ to $\mathcal{V}$.
4. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoKE}(\mathsf{u}, \hat{\mathsf{u}}_i; s_i)$, and $\mathcal{V}$ output 1 if and only if all PoKE accept and $\mathit{\Pi}_{i\in[n]}\hat{\mathsf{u}}_i = \mathsf{zw}^c$.

Fig. 4: Protocol $\mathsf{ZK}_{multiDL}$[BBF19]

Lemma 5. *In the generic group model,* Protocol $\mathsf{ZK}_{multiDL}$ *is an honest-verifier statistically zero-knowledge protocol for $\mathcal{R}_{multiDL}$, achieving a weak knowledge soundness defined as follows: There exists an extractor such that for any prover $\mathcal{P}^*$ convincing the verifier of statement $(\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w})$ with inverse-polynomial probability, the extractor can extract $(\{x'_i\}_{i\in[n]}, t)$ within an expected polynomial time such that $|t| \leq 2^\lambda$ and $\mathit{\Pi}_{i\in[n]}\mathsf{u}_i^{x'_i} = \mathsf{w}^t$.*

proof sketch. The honest-verifier statistically zero-knowledge property can be proved in the same manner as [BBF19] and the weak knowledge soundness follows Lemma.1 (the extension version for multidimensional DL) and the argument of knowledge property of PoKE (used for extracting s_i) directly.

A complete proof is shown in Appendix.B.

In the remainder of this paper, we only use the protocol $\mathsf{ZK}_{multiDL}$ for the cases when $n = 1$ or 2. For convinience, we shall refer to these protocols as ZK_{DL}

and ZK_{2DL}. Additionally, we denote their non-interactive versions obtained via the Fiat-Shamir heuristic as NIZK_{DL} and NIZK_{2DL}.

Standard zero-knowledge sets. Here we introduce the construction of standard ZKS in [PX09, XLL07, XLL08], but with some modifications. Instead of using RSA groups, we now use general groups of unknown orders. Furthermore, we use ZK_{DL} and ZK_{2DL} as sub-routine zero-knowledge protocol.

Let $\mathcal{H}_{prime}$ be a hash function that upon inputting a string outputs a large prime. And let $\mathcal{R}_{DL} = \{(\mathsf{u}, \mathsf{w}; x) | \mathsf{u}^x = \mathsf{w}\}$, $\mathcal{R}_{2DL} = \{(\mathsf{u}_1, \mathsf{u}_2, \mathsf{w}; x_1, x_2) | {\mathsf{u}_1}^{x_1} {\mathsf{u}_2}^{x_2} = \mathsf{w}\}$. The modified construction is shown below.

Standard zero-knowledge set scheme

$\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, Setup generates the description of an unknown-order group $\mathbb{G} \leftarrow GGen(\lambda)$ and a random group element $\mathsf{g} \xleftarrow{\$} \mathbb{G}$. Let B be the upper bound of $\mathbb{G}$ (i.e., $B \geq |\mathbb{G}|$). Sample the description of a hash function $\mathcal{H}_{prime}$ that on input a string, outputs a random prime larger than $2^{8\lambda} B^3$. Output CRS $\delta = (\mathbb{G}, \mathsf{g}, B, \mathcal{H}_{prime})$.

$\mathsf{Commit}(\delta, S)$: On input the set $S = \{x_1, \cdots, x_m\}$, Commit hashes them into large primes, i.e., for $i \in [m]$, $p_i \leftarrow \mathcal{H}_{prime}(x_i)$. Then Commit samples $r \xleftarrow{\$} [2^\lambda B]$, and outputs the commitment $\mathsf{C} = \mathsf{g}^{r \Pi_{i \in [m]} p_i}$ and the open information $\tau = (r, p_1, \cdots, p_m, S)$.

$\mathsf{Prove}(\delta, (C, \tau), x, S(x))$: Parse the input τ as $(r, p_1, \cdots, p_m, S)$.

a) If $S(x) = 1$, which means that $x \in S$, $p = \mathcal{H}_{prime}(x) \in \{p_1, \cdots, p_m\}$, Prove outputs the proof $\pi \leftarrow \mathsf{NIZK}_{DL}(\mathsf{g}^p, \mathsf{C}; r\Pi_{i \in [m]} p_i / p)$.

b) If $S(x) = \perp$, which means that $x \notin S$, $p = \mathcal{H}_{prime}(x) \notin \{p_1, \cdots, p_m\}$ and $\gcd(p, r\Pi_{i \in [m]} p_i) = 1$, Prove finds a, b such that $ap + br\Pi_{i \in [m]} p_i = 1$ and outputs $\pi \leftarrow \mathsf{NIZK}_{2DL}(\mathsf{g}^p, \mathsf{C}, \mathsf{g}; a, b)$.

$\mathsf{Verify}(\delta, \mathsf{C}, x, S(x), \pi)$: If $S(x) = 1$, check whether π is a valid NIZK_{DL} proof for statement $(\mathsf{g}^p, \mathsf{C}) \in \mathcal{R}_{DL}$. If $S(x) = \perp$, check whether π is a valid NIZK_{2DL} proof for statement $(\mathsf{g}^p, \mathsf{C}, \mathsf{g}) \in \mathcal{R}_{2DL}$. Verify outputs 1 if the check passes and outputs 0 otherwise.

Fig. 5: Protocol ZKS

Theorem 2. *The protocol constructed in Fig.5 is a ZKS scheme in the generic group model and random oracle model.*

The proof follows from [PX09, XLL07, XLL08]. We present a detailed proof in Appendix.C.

Remark 1. One can use the batch technique put forward in [BBF19] to batch the (non-)membership proofs. For example, to prove that $x'_1, \cdots, x'_t \in S$, the prover hashes them into primes $p'_1, \cdots p'_t$ by $\mathcal{H}_{prime}$ and then generates the proof $\pi \leftarrow \mathsf{NIZK}_{DL}(\mathsf{g}^{\Pi_{i\in[t]}p'_i}, \mathsf{C}; r\Pi_{i\in[m]}p_i/\Pi_{i\in[t]}p'_i)$. To prove that $x'_1, \cdots, x'_t \notin S$, the prover hashes them into primes $p'_1, \cdots p'_t$ by $\mathcal{H}_{prime}$ and finds $a, b \in \mathbb{Z}$ such that $a\Pi_{i\in[t]}p'_i + br\Pi_{i\in[m]}p_i = 1$, and then outputs $\pi \leftarrow \mathsf{NIZK}_{2DL}(\mathsf{g}^{\Pi_{i\in[t]}p'_i}, \mathsf{C}, \mathsf{g}; a, b)$.

3.2 Zero-Knowledge Protocols for Pseudo-Coprime Exponent Relation and Pseudo-DDH Relation

As shown in the technique overview subsection, due to the introduction of randomness in ZKS commitment, the basic set-operation relations on committed set can only be reduced to pseudo-coprime exponent relation and pseudo-DDH relation over group elements. Here pseudo-coprime exponent relation means that the greatest common divisor of the exponents of two group elements are small and pseudo-DDH relation means that a group element triple is close to a DDH tuple.

In this subsection, we construct two new zero-knowledge protocols for these two relations. Similar to Schnorr Σ-protocol, both protocols only provide a weak knowledge soundness, which are sufficient for our construction of ZKS with set-operation query.

Zero-knowledge protocol for pseudo-coprime exponents relation. Let u, v be ZKS commitments to sets X, Y respectively, i.e., $\mathsf{u} = \mathsf{g}^{r_u \Pi_{x\in X}\mathcal{H}_{prime}(x)}$, $\mathsf{v} = \mathsf{g}^{r_v \Pi_{x\in Y}\mathcal{H}_{prime}(x)}$. If $X \cap Y = \emptyset$, it yields that the exponents of u, v are almost coprime. We call such a tuple (u, v) as a pseudo-coprime exponents tuple and denote the **pseudo-coprime exponents relation** as follows:

$$\mathcal{R}_{coprime} = \left\{ (\mathsf{u}, \mathsf{v}, T; a_1, a_2) \middle| \begin{array}{ll} \gcd(a_1, a_2) \leq T & \wedge \\ \mathsf{u} = \mathsf{g}^{a_1}, \mathsf{v} = \mathsf{g}^{a_2} & \end{array} \right\}$$

We provide a zero-knowledge protocol for $\mathcal{R}_{coprime}$ in Fig.6. This protocol only satisfies a weak soundness that, a valid proof can convince the verifer that the statement belongs to a *relaxed* relation $\mathcal{R}^*_{coprime} = \{(\mathsf{u}, \mathsf{v}, T; t_1, t_2, c) | c \leq 2^{4\lambda}BT \wedge \mathsf{u}^{t_1}\mathsf{v}^{t_2} = \mathsf{g}^c\}$.

Lemma 6. *In the generic group model, $\mathsf{ZK}_{coprime}$ is an honest-verifier statistically zero-knowledge protocol for $\mathcal{R}_{coprime}$, achieving a weak knowledge soundness defined as follows: There exists an extractor such that for any prover P^* convincing the verifier with inverse-polynomial probability over statement $(\mathsf{u}, \mathsf{v}, T)$, the extractor can extract $t_1, t_2, c \in \mathbb{Z}$ within an expected polynomial time such that $|c| \leq 2^{4\lambda}BT$ and $\mathsf{u}^{t_1}\mathsf{v}^{t_2} = \mathsf{g}^c$.*

Proof. **Completeness** is obvious.

Weak knowledge soundness follows from the weak knowledge soundness of $\mathsf{ZK}_{boundedDL}$ and ZK_{2DL}. Specifically, the weak knowledge soundness of $\mathsf{ZK}_{boundedDL}$ allows us to extract $x, t \in \mathbb{Z}$ satisfying that $|x| \leq 2^{3\lambda}BT, |r'| \leq 2^{\lambda}$

Protocol $\mathsf{ZK}_{coprime}$ (Zero-knowledge protocol for $\mathcal{R}_{coprime}$)

Params: $\mathbb{G} \leftarrow GGen(\lambda), B \geq |\mathbb{G}|$; Common Input: $\mathsf{u}, \mathsf{v} \in \mathbb{G}, T \in \mathbb{Z}$;
Private Input for $\mathcal{P}$: $a_1, a_2 \in \mathbb{Z}$

1. $\mathcal{P}$ finds integers t_1, t_2 such that $t_1 a_1 + t_2 a_2 = \gcd(a_1, a_2)$. $\mathcal{P}$ samples $r \xleftarrow{\$} [2^\lambda B]$ and sends $\mathsf{Q} = \mathsf{g}^{r \gcd(a_1, a_2)} = \mathsf{u}^{rt_1} \mathsf{v}^{rt_2}$ to $\mathcal{V}$.
2. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{ZK}_{boundedDL}(\mathsf{g}, \mathsf{Q}, 2^\lambda BT; r \gcd(a_1, a_2))$ and $\mathsf{ZK}_{2DL}(\mathsf{u}, \mathsf{v}, \mathsf{Q}; rt_1, rt_2)$

Fig. 6: Protocol $\mathsf{ZK}_{coprime}$

and $\mathsf{Q}^{r'} = \mathsf{g}^x$. And the weak knowledge soundness of ZK_{2DL} enables the extraction of $t_1', t_2', r'' \in \mathbb{Z}$ such that $|r''| \leq 2^\lambda$ and $\mathsf{Q}^{r''} = \mathsf{u}^{t_1'} \mathsf{v}^{t_2'}$. With this, we obtain $\mathsf{u}^{r' t_1'} \mathsf{v}^{r' t_2'} = \mathsf{g}^{r'' x}$. By setting $t_1 = r' t_1', t_2 = r' t_2', c = r'' x$, we obtain $\mathsf{u}^{t_1} \mathsf{v}^{t_2} = \mathsf{g}^c$ and $|c| \leq 2^{4\lambda} BT$.

The simulator Sim of the **honest-verifier statistically zero-knowledge** property can be constructed as follows: Sim samples $r_1, r_2 \xleftarrow{\$} [2^\lambda B]$, sets $\mathsf{Q} = \mathsf{u}^{r_1} \mathsf{v}^{r_2}$, and then simulates the remaining zero-knowledge protocol to conclude the simulation.

Due to that both $\mathsf{ZK}_{boundedDL}$ and ZK_{2DL} are honest-verifier statistically zero-knowledge, we only need to prove that the distributions of $(\mathsf{u}, \mathsf{v}, \mathsf{Q})$ generated by the simulator and the honest prover are statistically indistinguishable. In other words, we need to show the following: For any fixed $\mathsf{u} = \mathsf{g}^{a_1}, \mathsf{v} = \mathsf{g}^{a_2}$, the statistical distance of the distributions $\{\mathsf{g}^{r \gcd(a_1, a_2)} | r \xleftarrow{\$} [2^\lambda B]\}$ and $\{\mathsf{u}^{r_1} \mathsf{v}^{r_2} = \mathsf{g}^{r_1 a_1 + r_2 a_2} | r_1, r_2 \xleftarrow{\$} [2^\lambda B]\}$ is exponentially small.

Let b be the order of $\mathsf{g}^{\gcd(a_1, a_2)}$, i.e., $b = \mathsf{Ord}(\mathsf{g}^{\gcd(a_1, a_2)}) \leq B$. From Lemma.2, $\{r \bmod b | r \xleftarrow{\$} [2^\lambda B]\}$ is exponentially close to the uniform distribution over $\mathbb{Z}_b$. Therefore the distribution $\{\mathsf{g}^{r \gcd(a_1, a_2)} | r \xleftarrow{\$} [2^\lambda B]\}$ is exponential close to the distribution $\{(\mathsf{g}^{\gcd(a_1, a_2)})^r | r \xleftarrow{\$} \mathbb{Z}_b\}$. From Lemma.3, $\{r_1 \cdot \frac{a_1}{\gcd(a_1, a_2)} + r_2 \cdot \frac{a_2}{\gcd(a_1, a_2)} \bmod b | r_1, r_2 \xleftarrow{\$} [2^\lambda B]\}$ is exponentially close to the uniform distribution over $\mathbb{Z}_b$. Therefore, we have that the distribution $\{\mathsf{g}^{r_1 a_1 + r_2 a_2} | r_1, r_2 \xleftarrow{\$} [2^\lambda B]\}$, which equals $\{(\mathsf{g}^{\gcd(a_1, a_2)})^{r_1 \cdot \frac{a_1}{\gcd(a_1, a_2)} + r_2 \cdot \frac{a_2}{\gcd(a_1, a_2)} \bmod b} | r_1, r_2 \xleftarrow{\$} [2^\lambda B]\}$, is also exponentially close to the distribution $\{(\mathsf{g}^{\gcd(a_1, a_2)})^r | r \xleftarrow{\$} \mathbb{Z}_b\}$. This concludes the lemma. □

Zero-knowledge protocol for pseudo-DDH relation. Let $\mathsf{u}, \mathsf{v}, \mathsf{w}$ be ZKS commitments to sets A, B, C, i.e., $\mathsf{u} = \mathsf{g}^{r_u \Pi_{x \in A} \mathcal{H}_{prime}(x)}$, $\mathsf{v} = \mathsf{g}^{r_v \Pi_{x \in B} \mathcal{H}_{prime}(x)}$, $\mathsf{w} = \mathsf{g}^{r_w \Pi_{x \in C} \mathcal{H}_{prime}(x)}$. If $C = A \cup B$ and $A \cap B = \emptyset$, it yields that the tuple $(\mathsf{u}, \mathsf{v}, \mathsf{w})$ is close to a DDH tuple. We call such a tuple $(\mathsf{u}, \mathsf{v}, \mathsf{w})$ as a pseudo-DDH tuple and denote the **pseudo-DDH relation** as follows:

$$\mathcal{R}_{pseudo-DDH} = \left\{ (\mathsf{u},\mathsf{v},\mathsf{w},T;x,y,a_1,a_2,a_3) \left| \begin{array}{ll} |a_1|,|a_2|,|a_3| \leq T & \wedge \\ \mathsf{gcd}(xy, \mathit{\Pi}_{i=1}^{2^\lambda|\mathbb{G}|} i) = 1 & \wedge \\ \mathsf{u} = \mathsf{g}^{a_1 x}, \mathsf{v} = \mathsf{g}^{a_2 y}, \mathsf{w} = \mathsf{g}^{a_3 xy} & \end{array} \right. \right\}.$$

In above relation, we require that the integers x and y are products of large primes (the second condition), which is necessary for our distance analysis in the proof of zero-knowledge property. We provide a zero-knowledge protocol for $\mathcal{R}_{pseudo-DDH}$ in Fig.7, which partially relies on Boneh et al's protocol. This protocol only satisfies a weak soundness that, a valid proof can convince the verifer that the statement belongs to a *relaxed* relation $\mathcal{R}^*_{pseudo-DDH} = \{(\mathsf{u},\mathsf{v},\mathsf{w},T;x,y,\{a_i,c_i\}_{i\in[3]})|\ |a_1|,|a_2|,|a_3| \leq 2^{4\lambda}BT, |c_1|,|c_2| \leq 2^{6\lambda}B^2, |c_3| \leq 2^{8\lambda}B^3 \wedge \mathsf{u}^{c_1} = \mathsf{g}^{a_1 x}, \mathsf{v}^{c_2} = \mathsf{g}^{a_2 y}, \mathsf{w}^{c_3} = \mathsf{g}^{a_3 xy}\}$.

Protocol $\mathsf{ZK}_{pseudo-DDH}$ (Zero-knowledge protocol for $\mathcal{R}_{pseudo-DDH}$)

Params: $\mathbb{G} \leftarrow GGen(\lambda), B \geq |\mathbb{G}|$; Common Input: $\mathsf{u},\mathsf{v},\mathsf{w} \in \mathbb{G}, T \in \mathbb{Z}$;
Private Input for $\mathcal{P}$: $a_1,a_2,a_3,x,y \in \mathbb{Z}$

1. $\mathcal{P}$ samples $r_1, r_2 \xleftarrow{\$} [2^{2\lambda}B]$ and sends $\mathsf{u}' = \mathsf{g}^{r_1 x}, \mathsf{v}' = \mathsf{g}^{r_2 y}, \mathsf{w}' = \mathsf{g}^{r_1 r_2 xy}$ to $\mathcal{V}$.
2. $\mathcal{V}$ sends $l_1 \xleftarrow{\$} \mathsf{Primes}(\lambda), l_2 \xleftarrow{\$} \mathsf{Primes}(\lambda)$.
3. $\mathcal{P}$ finds the quotient $q_1, q_2 \in \mathbb{Z}$ and residue $t_1 \in [l_1], t_2 \in [l_2]$ such that $r_1 x = q_1 l_1 + t_1$ and $r_2 y = q_2 l_2 + t_2$. $\mathcal{P}$ sends $\mathsf{Q}_1 = \mathsf{g}^{q_1}$, $\mathsf{Q}_1' = \mathsf{v}'^{q_1}$ and $\mathsf{Q}_2 = \mathsf{g}^{q_2}$, t_1 and t_2 to $\mathcal{V}$.
4. $\mathcal{V}$ checks $t_1 \in [l_1], t_2 \in [l_2]$, $\mathsf{Q}_1^{l_1}\mathsf{g}^{t_1} = \mathsf{u}'$, $\mathsf{Q}_1'^{l_1}\mathsf{v}'^{t_1} = \mathsf{w}'$, and $\mathsf{Q}_2^{l_2}\mathsf{g}^{t_2} = \mathsf{v}'$.
5. $\mathcal{P}$ samples $r_u, r_v, r_w \xleftarrow{\$} [2^{\lambda}B]$ and sends $\mathsf{u}'' = \mathsf{u}^{r_1 r_u}, \mathsf{v}'' = \mathsf{v}^{r_2 r_y}, \mathsf{w}'' = \mathsf{w}^{r_1 r_2 r_w}$ to $\mathcal{V}$.
6. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{ZK}_{boundedDL}(\mathsf{u},\mathsf{u}'',2^{3\lambda}B^2;r_1 r_u)$, $\mathsf{ZK}_{boundedDL}(\mathsf{u}',\mathsf{u}'',2^{\lambda}BT;a_1 r_u)$, $\mathsf{ZK}_{boundedDL}(\mathsf{v},\mathsf{v}'',2^{3\lambda}B^2;r_2 r_v)$, $\mathsf{ZK}_{boundedDL}(\mathsf{v}',\mathsf{v}'',2^{\lambda}BT;a_2 r_v)$, $\mathsf{ZK}_{boundedDL}(\mathsf{w},\mathsf{w}'',2^{5\lambda}B^3;r_1 r_2 r_w)$ and $\mathsf{ZK}_{boundedDL}(\mathsf{w}',\mathsf{w}'',2^{\lambda}BT;a_3 r_w)$.

Fig. 7: Protocol $\mathsf{ZK}_{pseudo-DDH}$

Lemma 7. *In the generic group model,* $\mathsf{ZK}_{pseudo-DDH}$ *is an honest-verifier statistically zero-knowledge protocol for* $\mathcal{R}_{pseudo-DDH}$*, achieving a weak knowledge soundness defined as follows: There exists an extractor such that for any prover* $\mathcal{P}^*$ *convincing the verifier with inverse-polynomial probability over statement* $(\mathsf{u},\mathsf{v},\mathsf{w},T)$*, the extractor can extract* $x,y,a_1,a_2,a_3,c_1,c_2,c_3 \in \mathbb{Z}$ *such that*

$|a_1|, |a_2|, |a_3| \leq 2^{4\lambda}BT$ $|c_1|, |c_2| \leq 2^{6\lambda}B^2$, $|c_3| \leq 2^{8\lambda}B^3$, *and* $\mathsf{u}^{c_1} = \mathsf{g}^{a_1 x}, \mathsf{v}^{c_2} = \mathsf{g}^{a_2 y}, \mathsf{w}^{c_3} = \mathsf{g}^{a_3 xy}$.

Proof. **Completeness** is obvious.

Weak knowledge soundness can be prove through the weak knowledge soundness of $\mathsf{ZK}_{boundedDL}$ and the knowledge extractor of PoKE. It is worth noting that steps 2 through 4 in our protocol are the same as in Boneh et al.'s PoDDH protocol [BBF19], which roughly consists of two PoKE protocols. We can use the knowledge extractor provided in [BBF19] to extract x, y satisfying $\mathsf{u}' = \mathsf{g}^x$, $\mathsf{v}' = \mathsf{g}^y$ and $\mathsf{w}' = \mathsf{g}^{xy}$. Meanwhile, the weak knowledge soundness of $\mathsf{ZK}_{boundedDL}$ allows us to extract a_u, c_u, a'_u, c'_u from the first two $\mathsf{ZK}_{boundedDL}$ protocols such that $|c_u|, |c'_u| \leq 2^\lambda, |a_u| \leq 2^{5\lambda}B^2, |a'_u| \leq 2^{3\lambda}BT$ and $\mathsf{u}^{a_u} = \mathsf{u}''^{c_u}, \mathsf{u}'^{a'_u} = \mathsf{u}''^{c'_u}$. Hence, we have $\mathsf{u}^{a_u c'_u} = \mathsf{g}^{a'_u c_u x}$. By setting $c_1 = a_u c'_u$, $a_1 = a'_u c_u$, we obtain $|c_1| \leq 2^{6\lambda}B^2, |a_1| \leq 2^{4\lambda}BT$ and $\mathsf{u}^{c_1} = \mathsf{g}^{a_1 x}$. Using the same strategy, we can extract c_2, c_3, a_2, a_3, thus meeting our goal.

The simulator Sim of **honest-verifier statistically zero-knowledge** property can be constructed as follows: Initially, Sim samples $r_1, r_2 \xleftarrow{\$} [2^{2\lambda}B]$ and sets $\mathsf{u}' = \mathsf{g}^{r_1}, \mathsf{v}' = \mathsf{g}^{r_2}, \mathsf{w}' = \mathsf{g}^{r_1 r_2}$. In step 3, Sim finds the quotient $q_1, q_2 \in \mathbb{Z}$ and residue t_1, t_2 such that $r_1 = q_1 l_1 + t_1$ and $r_2 = q_2 l_2 + t_2$, and then applies the same action as an honest prover. Subsequently, Sim samples $r_u, r_v, r_w \xleftarrow{\$} [2^\lambda B]$, sets $\mathsf{u}'' = \mathsf{u}^{r_1 r_u}, \mathsf{v}'' = \mathsf{v}^{r_2 r_v}, \mathsf{w}'' = \mathsf{w}^{r_1 r_2 r_w}$, and simulates the $\mathsf{ZK}_{boundedDL}$ protocols to conclude the simulation.

Due to the honest-verifier statistically zero-knowledge of $\mathsf{ZK}_{boundedDL}$, we only need to prove that for any fixed statement $(\mathsf{u}, \mathsf{v}, \mathsf{w}, T)$ and challenges l_1, l_2 (note that $l_1, l_2 \leq 2^\lambda$), the distributions of $(\mathsf{u}', \mathsf{v}', \mathsf{w}', \mathsf{Q}_1, \mathsf{Q}'_1, \mathsf{Q}_2, t_1, t_2, \mathsf{u}'', \mathsf{v}'', \mathsf{w}'')$ generated by the simulator and the honest prover are exponentially close. We denote these two distributions by $\mathcal{D}_{sim}$ and $\mathcal{D}_P$ respectively. Thus, for any fixed statement $(\mathsf{u}, \mathsf{v}, \mathsf{w}, T)$ and challenges l_1, l_2, we have the following:

$$\begin{aligned}
\mathcal{D}_{sim} = \{(&\mathsf{g}^{r_1}, \mathsf{g}^{r_2}, \mathsf{g}^{r_1 r_2}, \mathsf{g}^{\lfloor r_1/l_1 \rfloor}, (\mathsf{g}^{r_2})^{\lfloor r_1/l_1 \rfloor}, \mathsf{g}^{\lfloor r_2/l_2 \rfloor}, r_1 \mod l_1, \\
& r_2 \mod l_2, \mathsf{u}^{r_1 r_u}, \mathsf{v}^{r_2 r_v}, \mathsf{w}^{r_1 r_2 r_w}) | r_1, r_2 \xleftarrow{\$} [2^{2\lambda}B], r_u, r_v, r_w \xleftarrow{\$} [2^\lambda B]\} \\
\mathcal{D}_P = \{(&\mathsf{g}^{r_1 x}, \mathsf{g}^{r_2 y}, \mathsf{g}^{r_1 r_2 xy}, \mathsf{g}^{\lfloor r_1 x/l_1 \rfloor}, (\mathsf{g}^{r_2 y})^{\lfloor r_1 x/l_1 \rfloor}, \mathsf{g}^{\lfloor r_2 y/l_2 \rfloor}, r_1 x \mod l_1, \\
& r_2 y \mod l_2, \mathsf{u}^{r_1 r_u}, \mathsf{v}^{r_2 r_v}, \mathsf{w}^{r_1 r_2 r_w}) | r_1, r_2 \xleftarrow{\$} [2^{2\lambda}B], r_u, r_v, r_w \xleftarrow{\$} [2^\lambda B]\}
\end{aligned}$$

Denote $f(r_1, r_2, r_u, r_v, r_w) := (\mathsf{g}^{r_1}, \mathsf{g}^{r_2}, \mathsf{g}^{r_1 r_2},\ \mathsf{g}^{\lfloor r_1/l_1 \rfloor},\ (\mathsf{g}^{r_2})^{\lfloor r_1/l_1 \rfloor}, \mathsf{g}^{\lfloor r_2/l_2 \rfloor}, r_1 \mod l_1, r_2 \mod l_2, \mathsf{u}^{r_1 r_u}, \mathsf{v}^{r_2 r_v}, \mathsf{w}^{r_1 r_2 r_w})$. As a key observation, $f(r_1, r_2, r_u, r_v, r_w) = f(r_1 \mod l_1|\mathbb{G}|, r_2 \mod l_2|\mathbb{G}|, r_u \mod |\mathbb{G}|, r_v \mod |\mathbb{G}|, r_w \mod |\mathbb{G}|)$. We thus have that $\mathcal{D}_{sim} = \{f(r_1 \mod l_1|\mathbb{G}|, r_2 \mod l_2|\mathbb{G}|, r_u \mod |\mathbb{G}|, r_v \mod |\mathbb{G}|, r_w \mod |\mathbb{G}|) | r_1, r_2 \xleftarrow{\$} [2^{2\lambda}B], r_u, r_v, r_w \xleftarrow{\$} [2^\lambda B]\}$. Let x' (resp. y') be the element in $\mathbb{Z}_{|\mathbb{G}|}$ satisfying $xx' \equiv 1 \mod |\mathbb{G}|$ (resp. $yy' \equiv 1 \mod |\mathbb{G}|$). This is possible due to $\gcd(xy, |\mathbb{G}|) = 1$. Then, we obtain that $\mathcal{D}_P = \{f(r_1 x \mod l_1|\mathbb{G}|, r_2 y \mod l_1|\mathbb{G}|, r_u x' \mod |\mathbb{G}|, r_v y' \mod |\mathbb{G}|, r_w x' y' \mod |\mathbb{G}|) | r_1, r_2 \xleftarrow{\$} [2^{2\lambda}B], r_u, r_v, r_w \xleftarrow{\$} [2^\lambda B]\}$. From Lemma.2 and the fact that $\gcd(x, l_1|\mathbb{G}|) = 1, \gcd(y, l_2|\mathbb{G}|) = 1$, we have that both of the distributions $\mathcal{D}_{sim}$ and $\mathcal{D}_P$ are exponentially close to the distribution

$\{f(r_1, r_2, r_u, r_v,\ r_w)|r_1 \xleftarrow{\$} \mathbb{Z}_{l_1|\mathbb{G}|},\ r_2 \xleftarrow{\$} \mathbb{Z}_{l_2|\mathbb{G}|}, r_u, r_v, r_w \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|}\}$, which concludes the proof. □

Note that boths of the protocols provided in this subsection are constant-round public-coin protocols. One can use the Fiat-Shamir heuristic to obtain the non-interactive version of these zero-knowledge protocols. These non-interactive protocols satisfy zero-knowledge property and the same weak knowledge soundness property in the random oracle model. We denote these two non-interactive protocols as $\mathsf{NIZK}_{coprime}$ and $\mathsf{NIZK}_{pseudo-DDH}$.

4 Zero-Knowledge Set with Set-Operation Queries

In this section, we introduce the notion of ZKS with set-operation queries, which is the key ingredient for achieving our end goal of ZK-FEDBs. Moreover, we provide a concrete construction of ZKS with set-operation queries based on groups of unknown orders.

4.1 Definition

Informally, ZKS with set-operation queries allow one to commit to several sets $\{S_i\}_{i\in[m]}$, and then convincingly answer a) the (non-)membership queries and b) for any combined operation $\mathcal{Q}$ represented as a "circuit" of unions, intersections and set-differences, the queries in the form as "send me all records x in $\mathcal{Q}(S_1, \cdots, S_m)$," without revealing any extra knowledge. The formal definition of ZKS with set-operation queries is as follows:

Definition 2 (ZKS with Set-Operation Queries). *A ZKS with set-operation queries consists of six algorithms,* (Setup, Com, Prove, Verify, SO.Prove, SO.Verify)*, where* Setup, Com, Prove, Verify *are in the same form as a standard ZKS and*

- $\pi \leftarrow \mathsf{SO.Prove}(\delta, \widetilde{com}, \widetilde{\tau}, \mathcal{Q}, S_{output})$*: On input the CRS δ, the list of set commitments and the associated opening information $\widetilde{com} = (com_1, \cdots, com_m)$, $\widetilde{\tau} = (\tau_1, \cdots, \tau_m)$ where $(com_i, \tau_i) \in \mathsf{Com}(\delta, S_i)$, a combined operation $\mathcal{Q}$, and the output set S_{output},* SO.Prove *outputs a proof π of $S_{output} = \mathcal{Q}(S_1, \cdots, S_m)$.*
- $0/1 \leftarrow \mathsf{SO.Verify}(\delta, \widetilde{com}, \mathcal{Q}, S_{output}, \pi)$*: On input the CRS δ, the list of commitments $\widetilde{com}$, the combined operation $\mathcal{Q}$, the output S_{output}, and the proof π,* SO.Verify *either outputs* 1 *(denoting accept) or* 0 *(denoting reject).*

and satisfies the following properties:

- **Completeness:** *Completeness consists of two parts,*
  a) For any set S and any x,

$$\Pr\left[\mathsf{Verify}(\delta, com, x, S(x), \pi) = 1 \,\middle|\, \begin{array}{c}\delta \leftarrow \mathsf{Setup}(1^\lambda); (com, \tau) \leftarrow \mathsf{Com}(\delta, S);\\ \pi \leftarrow \mathsf{Prove}(\delta, com, \tau, x, S(x))\end{array}\right] = 1$$

b) For any sets $\{S_i\}_{i\in[m]}$ and any combined operation $\mathcal{Q}$ which takes m sets as input and outputs one set, let $S_{output} = \mathcal{Q}(S_1, \cdots, S_m)$, and thus

$$\Pr\left[\begin{array}{r}\mathsf{SO.Verify}(\delta, \widetilde{com}, \mathcal{Q}, \\ S_{output}, \pi) = 1\end{array}\middle|\begin{array}{c}\delta \leftarrow \mathsf{Setup}(1^\lambda); \\ \forall i \in [m], (com_i, \tau_i) \leftarrow \mathsf{Com}(\delta, S_i); \\ \pi \leftarrow \mathsf{SO.Prove}(\delta, \widetilde{com}, \widetilde{\tau}, \mathcal{Q}, S_{output})\end{array}\right] = 1$$

where $\widetilde{com} = (com_1, \cdots, com_m)$ and $\widetilde{\tau} = (\tau_1, \cdots, \tau_m)$.

- **Function Binding:** *For any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible:*
  1. *The challenger generates a CRS δ by running $\mathsf{Setup}(1^\lambda)$ and gives δ to the adversary $\mathcal{A}$.*
  2. *The adversary $\mathcal{A}$ outputs a set of commitments $\{com_i\}_{i\in[m]}$ and the following tuples:*
     (a) A series of (non-)membership query-proof tuples $\{(com'_j, x_j, v_j, \pi_j)\}_{j\in[n_1]}$, where $com'_j \in \{com_i\}_{i\in[m]}$ (supposing that $com'_j = com_{t_j}$).
     (b) A series of set-operation query-proof tuples $\{(\widetilde{com}_j, \mathcal{Q}_j, S'_j, \pi'_j)\}_{j\in[n_2]}$, where $\widetilde{com}_j$ is a list of commitments contained in $\{com_i\}_{i\in[m]}$ (supposing that $\widetilde{com}_j = (com_{t_{j1}}, com_{t_{j2}}, \cdots)$).
  3. *The adversary $\mathcal{A}$ wins the game if the following hold:*
     (a) For each $j \in [n_1]$, $\mathsf{Verify}(com'_j, x_j, v_j, \pi_j) = 1$
     (b) For each $j \in [n_2]$, $\mathsf{SO.Verify}(\widetilde{com}_j, \mathcal{Q}_j, S'_j, \pi'_j) = 1$.
     *(c) There do **not** exist sets $\{S_i\}_{i\in[m]}$ satisfying $S_{t_j}(x_j) = v_j$ for each $j \in [n_1]$ and $\mathcal{Q}_j(\widetilde{S}_j) = S'_j$ for each $j \in [n_2]$, where $\widetilde{S}_j = (S_{t_{j1}}, S_{t_{j2}}, \cdots)$.*
- **Zero-Knowledge:** *There exists a simulator Sim such that for any PPT adversary $\mathcal{A}$, the absolute value of the difference*

$$\Pr\left[\mathcal{A}^{\mathcal{O}_P}(\delta, st_{\mathcal{A}}, \{com_i\}_{i\in[m]}) = 1 \middle|\begin{array}{c}\delta \leftarrow \mathsf{Setup}(1^\lambda), \\ (\{S_i\}_{i\in[m]}, st_{\mathcal{A}}) \leftarrow \mathcal{A}(\delta), \\ \mathit{for}\ i \in [m], (com_i, \tau_i) \leftarrow \mathsf{Com}(\delta, S_i)\end{array}\right] -$$

$$\Pr\left[\mathcal{A}^{\mathcal{O}_S}(\delta, st_{\mathcal{A}}, \{com_i\}_{i\in[m]}) = 1 \middle|\begin{array}{c}(\delta, st_\delta) \leftarrow Sim(1^\lambda), \\ (\{S_i\}_{i\in[m]}, st_{\mathcal{A}}) \leftarrow \mathcal{A}(\delta), \\ (\{com_i\}_{i\in[m]}, st_S) \leftarrow Sim(\delta, m, st_\delta)\end{array}\right]$$

is negligible in λ, where $\mathcal{O}_P$ and $\mathcal{O}_S$ are defined as follows:

$\mathcal{O}_P$: On input (com_i, x) for some $i \in [m]$, $\mathcal{O}_P$ outputs $\pi \leftarrow \mathsf{Prove}(\delta, com_i, \tau_i, x, S_i(x))$. On input $(\widetilde{com}, \mathcal{Q})$ where $\widetilde{com} = (com_{t_1}, \cdots, com_{t_n})$ for some $t_1, \cdots, t_n \in [m]$, $\mathcal{O}_P$ outputs $\pi \leftarrow \mathsf{SO.Prove}(\delta, \widetilde{com}, \widetilde{\tau}, \mathcal{Q}, \mathcal{Q}(S_{t_1}, \cdots, S_{t_n}))$ where $\widetilde{\tau} = (\tau_{t_1}, \cdots, \tau_{t_n})$. In other cases, $\mathcal{O}_P$ outputs $\perp$.

$\mathcal{O}_S$: On input (com_i, x) for some $i \in [m]$, $\mathcal{O}_S$ outputs $\pi \leftarrow Sim(\delta, com, st_S, x, S_i(x))$. On input $(\widetilde{com}, Q)$ where $\widetilde{com} = (com_{t_1}, \cdots, com_{t_n})$ for some $t_1, \cdots, t_n \in [m]$, $\mathcal{O}_S$ outputs $\pi \leftarrow Sim(\delta, \widetilde{com}, st_S, \mathcal{Q}, \mathcal{Q}(S_{t_1}, \cdots, S_{t_n}))$. In other cases, $\mathcal{O}_S$ outputs $\perp$.

4.2 Construction of ZKS with Set-Operation Queries

This subsection describes our construction of a ZKS scheme with set-operation queries. Our construction builds on the standard ZKS described in Section.3.1. The Setup,Commit,Prove,Verify algorithms remain the same as Fig.5, and we only show how to construct $\mathsf{SO.Prove}$ and $\mathsf{SO.Verify}$ algorithms.

Combined Operations. In this paper, we denote a combined operation $\mathcal{Q}$ by a "circuit" of intersection "$\cap$", union "$\cup$", and set-difference "$\backslash$". Firstly, we demonstrate how to prove a basic set operation (namely the intersection, union or set-difference) on committed sets.

Algorithm for Intersection. Here we present a non-interactive protocol to prove that a commitment C_I commits to the intersection of two sets committed in C_{S_1} and C_{S_2}. Our protocol satisfies a special-purpose knowledge soundness, which roughly ensures the following:

- If one can generate a membership proof showing that x belongs to the set committed in C_I, then the extractor can generate membership proofs showing that x belongs to the sets committed in C_{S_1} and C_{S_2}.
- If one can generate a non-membership proof showing that x does not belong to the set committed in C_I, then the extractor can generate a non-membership proof showing that x does not belong to either the set committed in C_{S_1} or the set committed in C_{S_2}

Follow the fact that, for any S_1 and S_2, proving $I = S_1 \cap S_2$ only requires to show that I is a subset of S_1 and S_2, and $J_1 = S_1 \backslash I, J_2 = S_2 \backslash I$ are disjointed. We construct the zero-knowledge protocol $\mathsf{Intersection\text{-}NIZK}$ shown in Fig. 8. For any set $S = \{x_1, \cdots, x_m\}$, we denote by $\mathcal{H}_{prime}(S) = \{\mathcal{H}_{prime}(x_1), \cdots, \mathcal{H}_{prime}(x_m)\}$.

Lemma 8. $\mathsf{Intersection\text{-}NIZK}$ *is a zero-knowledge protocol, achieving a special purpose knowledge soundness defined as follows: There exists an extractor $E = (E_1, E_2)$ such that for any prover $\mathcal{P}^*$ convincing the verifier with inverse-polynomial probability over input $(\delta, \mathsf{C}_{S_1}, \mathsf{C}_{S_2}, \mathsf{C}_I)$, $E_1^{\mathcal{P}^*}$ can extract w in expected polynomial time such that the following holds:*

1. *On input w, $\mathsf{g}_a \in \mathbb{G}$ and prime $p \in \mathbb{Z}$ satisfying $p \geq 2^{8\lambda} B^3$ and $\mathsf{C}_I = \mathsf{g}_a^p$, $E_2(w, (\mathsf{g}_a, p))$ can output g_b and g_c such that $\mathsf{C}_{S_1} = \mathsf{g}_b^p$ and $\mathsf{C}_{S_2} = \mathsf{g}_c^p$.*
2. *On input w and $a, b, p \in \mathbb{Z}$ such that prime $p \geq 2^{8\lambda} B^3$ and $\mathsf{C}_I^a \cdot \mathsf{g}^{bp} = \mathsf{g}$, $E_2(w, (a, b, p))$ can output $a', b' \in \mathbb{Z}$ such that $\mathsf{C}_{S_1}^{a'} \cdot \mathsf{g}^{b'p} = \mathsf{g}$ or $\mathsf{C}_{S_2}^{a'} \cdot \mathsf{g}^{b'p} = \mathsf{g}$.*

Proof. **Completeness** directly follows the stucture of ZKS commitment, therefore we skip it here.

The simulator of the **zero-knowledge** property only needs to generate $\mathsf{C}_{J_1} = \mathsf{g}^{r_1}$ and $\mathsf{C}_{J_2} = \mathsf{g}^{r_2}$ by sampling $r_1, r_2 \xleftarrow{\$} [2^\lambda B]$, and generate π_1, π_2, π_3 using the simulator of $\mathsf{NIZK}_{pseudo-DDH}$ and $\mathsf{NIZK}_{coprime}$. Then the zero-knowledge property follows from the fact that the distributions of $\mathsf{C}_{J_1}, \mathsf{C}_{J_2}$ generated by the simulator are statistically indistinguishable from those generated by an honest prover and the zero-knowledge property of $\mathsf{NIZK}_{pseudo-DDH}$ and $\mathsf{NIZK}_{coprime}$.

Protocol Intersection-NIZK

Common Input: $\delta = (\mathbb{G}, \mathsf{g}, B, \mathcal{H}_{prime})$, $\mathsf{C}_{S_1}, \mathsf{C}_{S_2}, \mathsf{C}_I$;
Private Input for Prover: $S_1, S_2, I, \tau_{S_1}, \tau_{S_2}, \tau_I$ satisfying that
$(\mathsf{C}_{S_1}, \tau_{S_1}) \in \mathsf{Commit}(\delta, S_1)$; $(\mathsf{C}_{S_2}, \tau_{S_2}) \in \mathsf{Commit}(\delta, S_2)$;
$(\mathsf{C}_I, \tau_I) \in \mathsf{Commit}(\delta, I)$ and $S_1 \cap S_2 = I$.

Prover:

1. Generate the commitments C_{J_1} and C_{J_2} to the sets $J_1 = S_1 \backslash I$ and $J_2 = S_2 \backslash I$.
2. Run $\pi_1 \leftarrow \mathsf{NIZK}_{pseudo-DDH}(\mathsf{C}_I, \mathsf{C}_{J_1}, \mathsf{C}_{S_1}, 2^\lambda B)$ and $\pi_2 \leftarrow \mathsf{NIZK}_{pseudo-DDH}(\mathsf{C}_I, \mathsf{C}_{J_2}, \mathsf{C}_{S_2}, 2^\lambda B)$ using the required witness contained in the opening information.
3. Run $\pi_3 \leftarrow \mathsf{NIZK}_{coprime}(\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, 2^\lambda B)$ using the required witness contained in the opening information.
4. Output $\pi = (\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, \pi_1, \pi_2, \pi_3)$

Verifier: Output 1 iff. the proofs contained in π are valid.

Fig. 8: Protocol Intersection-NIZK

The proof of the **special purpose knowledge soundness** is as follows.

From the weak knowledge soundness of $\mathsf{NIZK}_{pseudo-DDH}$, E_1 can extract $w_1 = (x, y, a_1, a_2, a_3, c_1, c_2, c_3)$ such that $|a_1|, |a_2|, |a_3| \leq 2^{5\lambda} B^2$, $|c_1|, |c_2| \leq 2^{6\lambda} B^2$, $|c_3| \leq 2^{8\lambda} B^3$, and $\mathsf{C}_I^{c_1} = \mathsf{g}^{a_1 x}, \mathsf{C}_{J_1}^{c_2} = \mathsf{g}^{a_2 y}, \mathsf{C}_{S_1}^{c_3} = \mathsf{g}^{a_3 xy}$; and $w_2 = (x', y', a_1', a_2', a_3', c_1', c_2', c_3')$ such that $|a_1'|, |a_2'|, |a_3'| \leq 2^{5\lambda} B^2$, $|c_1'|, |c_2'| \leq 2^{6\lambda} B^2$, $|c_3'| \leq 2^{8\lambda} B^3$, and $\mathsf{C}_I^{c_1'} = \mathsf{g}^{a_1' x'}, \mathsf{C}_{J_2}^{c_2'} = \mathsf{g}^{a_2' y'}, \mathsf{C}_{S_2}^{c_3'} = \mathsf{g}^{a_3' x' y'}$. From the weak knowledge soundness of $\mathsf{NIZK}_{coprime}$, E_1 can extract $w_3 = (t_1, t_2, c)$ such that $|c| \leq 2^{5\lambda} B^2$ and $\mathsf{C}_{J_1}^{t_1} \mathsf{C}_{J_2}^{t_2} = \mathsf{g}^c$. Here, E_1 outputs $w = (\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, w_1, w_2, w_3)$.

Nextly, we show how E_2 works to conclude the proof:

1. On input w, $\mathsf{g}_a \in \mathbb{G}$ and prime $p \in \mathbb{Z}$ satisfying $p \geq 2^{8\lambda} B^3$ and $\mathsf{C}_I = \mathsf{g}_a^p$, E_2 firstly parses w as $(\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, w_1, w_2, w_3)$ and parses w_1 as $(x, y, a_1, a_2, a_3, c_1, c_2, c_3)$. Since $\mathsf{C}_I^{c_1} = \mathsf{g}^{a_1 x}$, it follows that $\mathsf{g}_a^{c_1 p} = \mathsf{g}^{a_1 x}$. We claim that $p | a_1 x$, otherwise, from Lemma.4, an attacker could easily find a p-root of g and break the strong RSA assumption. Since p is a prime larger than a_1, it follws that $p | x$ and $\mathsf{C}_{S_1}^{c_3} = \mathsf{g}^{a_3 xy} = (\mathsf{g}^{a_3 x_p y})^p$ where $x_p = x/p \in \mathbb{Z}$. As $\gcd(p, c_3) = 1$, from Lemma.4, E_2 can easily compute g_b from $(c_3, \mathsf{g}^{a_3 xy/p})$ such that $\mathsf{C}_{S_1} = \mathsf{g}_b^p$. Using the same strategy, E_2 can also compute g_c such that $\mathsf{C}_{S_2} = \mathsf{g}_c^p$.
2. On input w and $a, b, p \in \mathbb{Z}$ satisfying that p is a prime larger than $2^{8\lambda} B^3$ and $\mathsf{C}_I^a \cdot \mathsf{g}^{bp} = \mathsf{g}$, E_2 firstly parses w as $(\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, w_1, w_2, w_3)$ and further parses w_1 as $(x, y, a_1, a_2, a_3, c_1, c_2, c_3)$. We claim that $\gcd(p, x) = 1$, since it is known that $\mathsf{C}_I^a \cdot \mathsf{g}^{bp} = \mathsf{g}$ and $\mathsf{C}_I^{c_1} = \mathsf{g}^{a_1 x}$; otherwise, an attacker could easily break the strong RSA assumption. If $\gcd(p, y) = 1$ (or equivalently, $\gcd(p, a_3 xy) = 1$), E_2 can easily find integers α, β satisfying $\alpha p + \beta a_3 xy = 1$, and output

$a' = \beta c_3$ and $b' = \alpha$ such that $\mathsf{C}_{S_1}^{a'} \cdot \mathsf{g}^{b'p} = \mathsf{C}_{S_1}^{\beta c_3} \mathsf{g}^{\alpha p} = \mathsf{g}^{\beta a_3 x y} \mathsf{g}^{\alpha p} = \mathsf{g}$. Similarly, parsing $w_2 = (x', y', a_1', a_2', a_3', c_1', c_2', c_3')$, if $\gcd(p, y') = 1$, E_2 can also compute $a', b' \in \mathbb{Z}$ such that $\mathsf{C}_{S_2}^{a'} \cdot \mathsf{g}^{b'p} = \mathsf{g}$. The construction of E_2 concludes by claiming that at least one of the $\gcd(p, y) = 1$, $\gcd(p, y') = 1$ must happen. Otherwise, an attacker could easily break the strong RSA assumption as follows: Since p is a large prime and $\gcd(p, y) \neq 1, \gcd(p, y') \neq 1$, it follows that $p | y$, $p | y'$. Then, from Lemma.4 and the fact $\mathsf{C}_{J_1}^{c_2} = \mathsf{g}^{a_2 y}, \mathsf{C}_{J_2}^{c_2'} = \mathsf{g}^{a_2' y'}$, it can compute $\mathsf{h}_1, \mathsf{h}_2$ such that $\mathsf{C}_{J_1} = \mathsf{h}_1^p$ and $\mathsf{C}_{J_2} = \mathsf{h}_2^p$. Subsequently, from Lemma.4 and the fact $\mathsf{C}_{J_1}^{t_1} \mathsf{C}_{J_2}^{t_2} = \mathsf{g}^c$ where $w_3 = (t_1, t_2, c)$, it can compute h such that $\mathsf{h}^p = \mathsf{g}$, breaking the strong RSA assumption.

□

Algorithm for Union. Herein we present a non-interactive protocol to prove that a commitment C_U commits to the union of two sets committed in C_{S_1} and C_{S_2}. Our protocol satisfies a special-purpose knowledge soundness, which roughly ensures the following:

- If one can generate a membership proof showing that x belongs to the set committed in C_U, the extractor can generate a membership proof showing that x belongs to the set committed in either C_{S_1} or C_{S_2}.
- If one can generate a non-membership proof showing that x does not belong to the set committed in C_I, then the extractor can generate non-membership proofs showing that x does not belong to the sets committed in C_{S_1} and C_{S_2}.

We construct the zero-knowledge protocol $\mathsf{Union\text{-}NIZK}$ in Fig. 9.

Lemma 9. $\mathsf{Union\text{-}NIZK}$ *is a zero-knowledge protocol, achieving a special purpose knowledge soundness defined as follows: There exists an extractor $E = (E_1, E_2)$ such that for any prover $\mathcal{P}^*$ convincing the verifier with inverse-polynomial probability over input $(\delta, \mathsf{C}_{S_1}, \mathsf{C}_{S_2}, \mathsf{C}_I)$, $E_1^{\mathcal{P}^*}$ can extract w within an expected time such that the following hold:*

1. *On input w and $\mathsf{g}_a \in \mathbb{Z}$ and prime p satisfying $p \geq 2^{8\lambda} B^3$ and $\mathsf{C}_U = \mathsf{g}_a^p$, $E_2(w, (\mathsf{g}^a, p))$ can output g_b such that $\mathsf{C}_{S_1} = \mathsf{g}_b^p$ or $\mathsf{C}_{S_2} = \mathsf{g}_b^p$.*
2. *On input w and $a, b, p \in \mathbb{Z}$ such that prime $p \geq 2^{8\lambda} B^3$ and $\mathsf{C}_U^a \cdot \mathsf{g}^{bp} = \mathsf{g}$, $E_2(w, (a, b, p))$ can output $a', b', a'', b'' \in \mathbb{Z}$ such that $\mathsf{C}_{S_1}^{a'} \cdot \mathsf{g}^{b'p} = \mathsf{g}$ and $\mathsf{C}_{S_2}^{a''} \cdot \mathsf{g}^{b''p} = \mathsf{g}$.*

The proof of this lemma is similar to Lemma.8. We defer it to Appendix.D.

Algorithm for Set-Difference. We present a non-interactive protocol to prove that the difference D between two sets S_1, S_2 committed in C_{S_1} and C_{S_2} (i.e., $D = S_1 \backslash S_2$) is committed in C_D. Our protocol satisfies a special-purpose knowledge soundness, which roughly ensures the following:

- If one can generate a membership proof showing that x belongs to the set committed in C_D, the extractor can generate a membership proof showing that x belongs to the set committed in C_{S_1} and a non-membership proof showing that x doesn't belong to the set committed in C_{S_2}.

Protocol Union-NIZK

Common Input: $\delta = (\mathbb{G}, \mathsf{g}, B, \mathcal{H}_{prime})$, $\mathsf{C}_{S_1}, \mathsf{C}_{S_2}, \mathsf{C}_U$;
Private Input for Prover: $S_1, S_2, U, \tau_{S_1}, \tau_{S_2}, \tau_U$ satisfying that $(\mathsf{C}_{S_1}, \tau_{S_1}) \in \mathsf{Commit}(\delta, S_1)$; $(\mathsf{C}_{S_2}, \tau_{S_2}) \in \mathsf{Commit}(\delta, S_2)$; $(\mathsf{C}_U, \tau_U) \in \mathsf{Commit}(\delta, U)$ and $S_1 \cup S_2 = U$.

Prover

1. Generates the commitment $\mathsf{C}_I, \mathsf{C}_{J_1}, \mathsf{C}_{J_2}$ to the sets $I = S_1 \cap S_2, J_1 = U \backslash S_1, J_2 = U \backslash S_2$.
2. Run $\pi_1 \leftarrow \mathsf{NIZK}_{pseudo-DDH}(\mathsf{C}_I, \mathsf{C}_{J_1}, \mathsf{C}_{S_2}, 2^\lambda B)$ using the required witness contained in the opening information.
3. Run $\pi_2 \leftarrow \mathsf{NIZK}_{pseudo-DDH}(\mathsf{C}_{S_1}, \mathsf{C}_{J_1}, \mathsf{C}_U, 2^\lambda B)$ and $\pi_3 \leftarrow \mathsf{NIZK}_{pseudo-DDH}(\mathsf{C}_{S_2}, \mathsf{C}_{J_2}, \mathsf{C}_U, 2^\lambda B)$ using the required witness contained in the opening information.
4. Output $\pi = (\mathsf{C}_I, \mathsf{C}_{J_1}, \mathsf{C}_{J_2}, \pi_1, \pi_2, \pi_3)$

Verifier: Output 1 iff. the proofs contained in π are valid.

Fig. 9: Protocol Union-NIZK

– If one can generate a non-membership proof showing that x does not belong to the set committed in C_D, the extractor can generate a non-membership proof showing that x does not belong to the set committed in C_{S_1} or a membership proof showing that x belongs to the set committed in C_{S_2}.

We construct the zero-knowledge protocol Difference-NIZK in Fig. 10.

Lemma 10. Difference-NIZK *is a zero-knowledge protocol, achieving a special purpose knowledge soundness defined as follows: There exists a extractor $E = (E_1, E_2)$ such that for any prover $\mathcal{P}^*$ convincing the verifier with inverse-polynomial probability over input $(\delta, \mathsf{C}_{S_1}, \mathsf{C}_{S_2}, \mathsf{C}_D)$, $E_1^{\mathcal{P}^*}$ can extract w such that the following holds:*

1. *On input w, $\mathsf{g}_a \in \mathbb{G}$ and prime $p \in \mathbb{Z}$ such that $p \geq 2^{8\lambda} B^3$ and $\mathsf{C}_D = \mathsf{g}_a^p$, $E_2(w, (\mathsf{g}_a, p))$ can output g_b and a, b such that $\mathsf{C}_{S_1} = \mathsf{g}_b^p$ and $\mathsf{C}_{S_2}^a \mathsf{g}^{pb} = \mathsf{g}$.*
2. *On input w and $a, b, p \in \mathbb{Z}$ such that prime $p \geq 2^{8\lambda} B^3$ and $\mathsf{C}_D^a \cdot \mathsf{g}^{bp} = \mathsf{g}$, $E_2(w, (a, b, p))$ can output g_a or a', b' such that $\mathsf{C}_{S_2} = \mathsf{g}_a^p$ or $\mathsf{C}_{S_1}^{a'} \cdot \mathsf{g}^{b'p} = g$.*

The proof of this lemma is similar to Lemma.8, and we defer it to Appendix.D.

Algorithm for Combined Operations. Using above algorithms, we now construct the SO.Prove and SO.Verify algorithms to conclude the construction of ZKS with set-operation queries. Remind that a combined operation $\mathcal{Q}$ is a circuit comprised of gates and wires. Each gate corresponds to an intersection, union, or set difference operation. Just like in Boolean circuits, when given a string as input, each wire in Boolean circuit has a deterministic bit in $\{0, 1\}$.

Protocol Difference-NIZK

Common Input: $\delta = (\mathbb{G}, \mathsf{g}, B, \mathcal{H}_{prime})$, $\mathsf{C}_{S_1}, \mathsf{C}_{S_2}, \mathsf{C}_D$;
Private Input for Prover: $S_1, S_2, S_D, \tau_{S_1}, \tau_{S_2}, \tau_D$ satisfying that $(\mathsf{C}_{S_1}, \tau_{S_1}) \in \mathsf{Commit}(\delta, S_1)$; $(\mathsf{C}_{S_2}, \tau_{S_2}) \in \mathsf{Commit}(\delta, S_2)$; $(\mathsf{C}_D, \tau_D) \in \mathsf{Commit}(\delta, D)$ and $S_1 \backslash S_2 = D$.

Prover:

1. Generate the commitments $\mathsf{C}_I, \mathsf{C}_J$ to the sets $I = S_1 \cap S_2, J = S_2 \backslash I$.
2. Run $\pi_1 \leftarrow \mathsf{NIZK}_{pseudo-DDH}(\mathsf{C}_I, \mathsf{C}_D, \mathsf{C}_{S_1}, 2^\lambda B)$ and $\pi_2 \leftarrow \mathsf{NIZK}_{pseudo-DDH}(\mathsf{C}_I, \mathsf{C}_J, \mathsf{C}_{S_2}, 2^\lambda B)$ using the required witness contained in the opening information.
3. Run $\pi_3 \leftarrow \mathsf{NIZK}_{coprime}(\mathsf{C}_D, \mathsf{C}_{S_2}, 2^\lambda B)$ using the required witness contained in the opening information.
4. Output $\pi = (\mathsf{C}_I, \mathsf{C}_J, \pi_1, \pi_2, \pi_3)$

Verifier: Output 1 iff. the proofs contained in π are valid.

Fig. 10: Protocol Difference-NIZK

Each wire in the set-operation circuit corresponds to a deterministic set when provided with a specific input.

Therefore, to prove that a combined operation $\mathcal{Q}$ on (committed) sets is performed honestly, we only need to show that each gate in $\mathcal{Q}$ is performed honestly, i.e., for each gate corresponding a basic set operation, the (committed) set corresponding to the output wire is the result of applying this set operation to the (committed) sets corresponding to the input wires.

Without loss of generality, assume that the gates of $\mathcal{Q}$ are numbered based on their execution order. That is to say, the input wires of the gate i are either the output wire of some gate $j < i$ or an input wire of $\mathcal{Q}$, and the output wire of the last gate is also the output wire of $\mathcal{Q}$. The protocol is shown in Fig.11:

Theorem 3. *The algorithms* (Setup, Com, Prove, Verify) *in Fig.5 together with the algorithms* (SO.Prove, SO.Verify) *in Fig.11 consist a ZKS with set-operation queries in the generic group model and random oracle model.*

Proof. **Completeness** is oblivious.

To prove the **function binding** property, we will show the existence of a extractor E satisfying that, for any PPT adversary $\mathcal{A}$ generating a series of valid query-proof tuples with a noticeable probability, E can either extract a series of sets satisfying all queries or break the strong RSA assumption. Here, E is constructed as follows.

1. First, E invokes $\mathcal{A}$ to obtain m commitments, $\mathsf{C}_1, \cdots, \mathsf{C}_m$. Then, E initializes $2m+1$ sets, labeled as $S_0, S_1, S_1', \cdots, S_m, S_m'$. Here, S_0 is used to record all the elements x appearing in the query-proof queries generated by $\mathcal{A}$ (including the elements in the output set of a set-operation query). In addition,

Algorithm for Verifiable Combined Operations

$\mathsf{SO.Prove}(\delta, \widetilde{com}, \widetilde{\tau}, \mathcal{Q}, S_{output})$: On input the CRS δ, the list of commitments and the associated opening information $\widetilde{com} = (\mathsf{C}_1, \cdots, \mathsf{C}_m)$, $\widetilde{\tau} = (\tau_1, \cdots, \tau_m)$ where $(\mathsf{C}_i, \tau_i) \in \mathsf{Com}(\delta, S_i)$, a combined operation $\mathcal{Q}$ and the target output set S_{output}. $\mathsf{SO.Prove}$ runs as follows.

1. Recover the sets $\{S_i\}_{i\in[m]}$ from the opening information. Regard $\mathcal{Q}$ as a "circuit" and let l be the number of gates. Run $\mathcal{Q}(S_1, \cdots, S_m)$ to obtain the sets $S'_1, \cdots, S'_l$ corresponding to the output wires of l gates in $\mathcal{Q}$ (note that $S'_l = S_{output}$). Generate the commitments $\mathsf{C}'_1, \cdots, \mathsf{C}'_l$ to the sets $S'_1, \cdots, S'_l$. (Note that the sets corresponding to the input wires of circuit are already committed by $\{\mathsf{C}_i\}_{i\in[m]}$.)
2. For each gate i, suppose S_{a_i}, S_{b_i} are the sets corresponding to its input wires, S_{c_i} is the set corresponding to its output wire and $\mathsf{C}_{a_i}, \mathsf{C}_{b_i}, \mathsf{C}_{c_i}$ are their commitments.
   (a) If the gate corresponds to "interaction", $\mathsf{SO.Prove}$ runs $\pi_i \leftarrow \mathsf{Intersection\text{-}NIZK}(\delta, \mathsf{C}_{a_i}, \mathsf{C}_{b_i}, \mathsf{C}_{c_i})$.
   (b) If the gate corresponds to "union", $\mathsf{SO.Prove}$ runs $\pi_i \leftarrow \mathsf{Union\text{-}NIZK}(\delta, \mathsf{C}_{a_i}, \mathsf{C}_{b_i}, \mathsf{C}_{c_i})$.
   (c) If the gate corresponds to "set-difference", $\mathsf{SO.Prove}$ runs $\pi_i \leftarrow \mathsf{Difference\text{-}NIZK}(\delta, \mathsf{C}_{a_i}, \mathsf{C}_{b_i}, \mathsf{C}_{c_i})$.
3. Output $\pi = (\mathsf{C}'_1, \cdots, \mathsf{C}'_l, \pi_1, \cdots, \pi_l, \tau'_l)$, where τ'_l is the opening information of C'_l.

$\mathsf{SO.Verify}(\delta, \widetilde{com}, \mathcal{Q}, S_{output}, \pi)$: Parse π as $(\mathsf{C}'_1, \cdots, \mathsf{C}'_l, \pi_1, \cdots, \pi_l, \tau'_l)$. Regard $\mathcal{Q}$ as a "circuit" in the same way as the prover. For each gate i, suppose $\mathsf{C}_{a_i}, \mathsf{C}_{b_i}$ are the commitments that commit to the sets corresponding to the input wires and C_{c_i} is the commitment that commits to the set corresponding to the output wire. If this gate corresponds to "interaction" (resp. "union", "set-difference"), check whether π_i is a valid $\mathsf{Intersection\text{-}NIZK}$ (resp. $\mathsf{Union\text{-}NIZK}$, $\mathsf{Difference\text{-}NIZK}$) proof over the statement $\delta, \mathsf{C}_{a_i}, \mathsf{C}_{b_i}, \mathsf{C}_{c_i}$. Use τ'_l to check whether C'_l is a commitment to the set S_{output}. Output 1 iff. all checks pass.

Fig. 11: Protocol for Verifiable Combined Operations

for any $i \in [m]$, S_i is the sets of elements that, believed by E, are contained in the set committed in C_i; S'_i is the sets of elements that, believed by E, are not contained in the set committed in C_i. Here, E invokes $\mathcal{A}$ to obtain the query-proof tuples and adds all appearing elements to S_0.

2. For each membership proof proving that x belongs to the set committed in C_i, E adds x to S_i, and extracts and records the tuple $(x, \mathsf{g}_x, p, \mathsf{C}_i)$ from the proof such that $\mathsf{g}_x^p = \mathsf{C}_i$ and $p = \mathcal{H}_{prime}(x)$. (The extraction of the

tuple is trivial). We call such a tuple as membership tuple. For each non-membership proof proving that x does not belong to the set committed in C_i, E adds x to S'_i, and extracts and records $(x, a, b, p, \mathsf{C}_i)$ such that $\mathsf{C}_i^a \mathsf{g}^{pb} = \mathsf{g}$ and $p = \mathcal{H}_{prime}(x)$. We call such a tuple as non-membership tuple. Furthmore, for each set-operation query-proof tuple, E uses the extractors E_1 of Intersection-NIZK, Union-NIZK and Difference-NIZK to extract the corresponding w.

3. For each element $x \in S_0$, E applies the following. For each set-operation query-proof tuple, if x belongs to the output set S_{output} whose commitment is C'_l, E can obtain a membership tuple $(x, \mathsf{g}_x, p, \mathsf{C}'_l)$ from proof. On the other hand, if x does not belong to the output set, then E can obtain a non-membership tuple $(x, a, b, p, \mathsf{C}'_l)$. According to the special purpose knowledge soundness of Intersection-NIZK, Union-NIZK, and Difference-NIZK, for each gate, when given a (non-)membership tuple associated to the output wires (we call a (non-)membership tuple associated to a wire if the commitment in this tuple is the one committing to the set corresponding to this wire), one can extract one or two (non-)membership tuples associated to the input wires. E can hence recursively obtain a series of tuples associated to input wires of the combined operation. As a result, for each obtained tuple of the form $(x, \mathsf{g}', p, \mathsf{C}_i)$, E adds x to S_i, and for each obtained tuple of the form $(x, a, b, p, \mathsf{C}_i)$, E adds x to S'_i.
4. If there are no contradictions (that is, there are no elements x and $i \in [m]$ such that $x \in S_i \wedge x \in S'_i$), then E outputs $S_1, \cdots, S_m$. Otherwise, it means that there exists $(x, \mathsf{g}_x, p, \mathsf{C}_i)$ and $(x, a, b, p, \mathsf{C}_i)$ such that $\mathsf{g}_x^p = \mathsf{C}_i$ and $\mathsf{C}_i^a \mathsf{g}^{pb} = \mathsf{g}$, breaking the strong RSA Assumption.

Now we only need to show that the sets $S_1, \cdots, S_m$ outputted by E satisfy all queries. From step 2, we can see that these sets already satisfy the (non-)membership queries. As for set-operation queries, we need to show that: For each set-operation query-proof $(\mathsf{C}_{t_1}, \cdots, \mathsf{C}_{t_k}, \mathcal{Q}, S_{output}, \pi)$, $\mathcal{Q}(S_{t_1}, \cdots, S_{t_k}) = S_{output}$.

Let l be the number of the gates of $\mathcal{Q}$, parse π as $(\mathsf{C}'_1, \cdots, \mathsf{C}'_l, \pi_1, \cdots, \pi_l, \tau'_l)$. Run $\mathcal{Q}(S_{t_1}, \cdots, S_{t_k})$ to obtain the sets $S'_1, \cdots, S'_l$ correspnding to the output wires of the gates in $\mathcal{Q}$ (thus, $S'_l = \mathcal{Q}(S_{t_1}, \cdots, S_{t_k})$). Denote by $(S''_1, \cdots, S''_{k+l}) = (S_{t_1}, \cdots, S_{t_k}, S'_1, \cdots, S'_l)$ and $(\mathsf{C}''_1, \cdots, \mathsf{C}''_{k+l}) = (\mathsf{C}_{t_1}, \cdots, \mathsf{C}_{t_k}, \mathsf{C}'_1, \cdots, \mathsf{C}'_l)$. Remind that E will extract lots of tuples associated to the wires in $\mathcal{Q}$ in step 3. Here, we recursively prove the following statements are true for each $i \in [k + l]$:

For each $x \in S_0$, if E used to extract a tuple of the form $(x, \mathsf{g}', p, \mathsf{C}''_i)$ in step 3, then $x \in S''_i$; And if E used to extract a tuple of the form $(x, a, b, p, \mathsf{C}''_i)$ in step 3, then $x \notin S''_i$.

Firstly, from the definition of $S_1, \cdots, S_m$, above statements are true for $i \in [k]$. Suppose above statements are true for $i \in [k + t - 1]$. Then for gate t with sets S''_{j_1}, S''_{j_2} $(j_1, j_2 \leq k + t - 1)$ corresponding to the input wires, if the gate corresponding to "interactive" and $(x, \mathsf{g}_a, p, \mathsf{C}''_{k+t})$ (resp. $(x, a, b, p, \mathsf{C}''_{k+t})$) is extracted, then from the knowledge soundness of Intersection-NIZK, $(x, \mathsf{g}_b, p, \mathsf{C}''_{j_1})$

and $(x, \mathsf{g}_c, p, \mathsf{C}''_{j_2})$ (resp. $(x, a', b', \mathsf{C}''_{j_1})$ or $(x, a', b', \mathsf{C}''_{j_2})$) are extracted by E. Since $j_1, j_2 \leq k + t - 1$, it follows that $x \in S''_{j_1}, x \in S''_{j_2}$ (resp. $x \notin S''_{j_1}$ or $x \notin S''_{j_2}$), and therefore $x \in S''_{k+t} = S''_{j_1} \cap S''_{j_2}$ (resp. $x \notin S''_{k+t} = S''_{j_1} \cap S''_{j_2}$). The case that gate t corresponds to "union" or "set-difference" can be proved similarly. Therefore above statement is also true for $i = k + t$. Recursively, the above statements are true for each $i \in [k + l]$. Note that S''_{k+l} is the output set. Remind that if $x \in S_{output}$ (resp. $x \notin S_{output}$), E will extract $(x, \mathsf{g}', p, \mathsf{C}''_{l+k})$ (resp. $(x, a, b, p, \mathsf{C}''_{l+k})$), which means that $x \in S''_{k+l}$ (resp. $x \notin S''_{m+l}$). Therefore we have $\mathcal{Q}(S_{t_1}, \cdots, S_{t_k}) = S'_l = S''_{k+l} = S_{output}$, which concludes the proof of function binding.

For the **zero-knowledge** property, due to that the distribution of the ZKS commitment is statistically indistinguishable from $\{\mathsf{g}^r | r \leftarrow [2^\lambda B]\}$, the simulator can sample element from $\{\mathsf{g}^r | r \leftarrow [2^\lambda B]\}$ as the commitments and then use the simulators of Intersection-NIZK, Union-NIZK and Difference-NIZK to conclude the simulations. □

Remark 2. One can use the randomness r'_l applied in the commitment C'_l to replace the opening information τ'_l, which is also sufficient to check whether C'_l is a commitment to S_{output}. Therefore the proof size of a set-operation query is only linear in the size of combined operation $\mathcal{Q}$ and the length of elements in S.

5 Zero-Knowledge Functional Elementary Databases

This section consists of three parts. Firstly, we discribe the difinition of ZK-FEDBs. Secendly, we show how to transform a Boolean circuit query into a set-operation query on related sets. Finally we show how to construct a ZK-FEDBs from standard ZE-EDBs and ZKS with set-operation queries.

5.1 Definition of Zero-Knowledge Functional Elementary Databases

Informally, a ZK-FEDB allows one to commit an elementary database D of key-value pairs (x, v) and then, for any Boolean circuit f, convincingly answer the queries in the form of "send me all records (x, v) in D satisfying $f(x, v) = 1$". Here we write the output database as $D(f)$ (i.e., $D(f) = \{(x, v) \in D | f(x, v) = 1\}$) and regard the membership queries (supported by the standard ZK-EDB) as a type of special function query.

Definition 3 (Zero-Knowledge Functional Elementary Databases). *A Zero-Knowledge Functional Elementary Database consists of four algorithms* (Setup, Com, Prove, Verify),

- $\delta \leftarrow \mathsf{Setup}(1^\lambda)$*: On input a security parameter* 1^λ*,* Setup *outputs a random string (or a structured reference string)* δ *as the CRS.*
- $(com, \tau) \leftarrow \mathsf{Com}(\delta, D)$*: On input a CRS* δ *and an elementary database* D*,* Com *outputs a commitment of database com and opening information* τ*.*

- $\pi \leftarrow \mathsf{Prove}(\delta, com, \tau, f, D_{output})$: *On input the CRS δ, the database commitment and the associated opening information (com, τ), a Boolean circuit f, and the target output D_{output},* Prove *outputs a proof π for $D_{output} = D(f)$.*
- $0/1 \leftarrow \mathsf{Verify}(\delta, com, f, D_{output}, \pi)$: *On input the CRS δ, the commitment com, the boolean circuit f, the target output D_{output} and the proof π,* Verify *accepts or rejects.*

It satisfies the following three properties:

- **Completeness:** *For any elementary database D and any Boolean circuit f,*

$$\Pr\left[\mathsf{Verify}(\delta, com, f, D(f), \pi) = 1 \,\middle|\, \begin{array}{c}\delta \leftarrow \mathsf{Setup}(1^\lambda); (com, \tau) \leftarrow \mathsf{Com}(\delta, D);\\ \pi \leftarrow \mathsf{Prove}(\delta, com, \tau, f, D(f))\end{array}\right] = 1$$

- **Function binding:** *For any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the following game is negligible:*
  1. *The challenger generates a CRS δ by running* $\mathsf{Setup}(1^\lambda)$ *and gives δ to adversary $\mathcal{A}$.*
  2. *The adversary $\mathcal{A}$ outputs a commitment com and a series of function query-proof tuples $\{(f_i, D_i, \pi_i)\}_{i\in[n]}$.*
  3. *The adversary $\mathcal{A}$ wins the game if the following hold: a) For each $i \in [n]$,* $\mathsf{Verify}(\delta, com, f_i, D_i, \pi_i) = 1$ *and b) there does not exist a database D satisfying $D(f_i) = D_i$ for each $i \in [n]$.*
- **Zero-Knowledge:** *There exists a simulator Sim such that for any PPT adversary $\mathcal{A}$, the absolute value of the difference*

$$\Pr\left[\mathcal{A}^{\mathcal{O}_P}(\delta, st_{\mathcal{A}}, com) = 1 \,\middle|\, \begin{array}{c}\delta \leftarrow \mathsf{Setup}(1^\lambda), (D, st_{\mathcal{A}}) \leftarrow \mathcal{A}(\delta),\\ (com, \tau) \leftarrow \mathsf{Com}(\delta, D)\end{array}\right] -$$
$$\Pr\left[\mathcal{A}^{\mathcal{O}_S}(\delta, st_{\mathcal{A}}, com) = 1 \,\middle|\, \begin{array}{c}(\delta, st_\delta) \leftarrow Sim(1^\lambda), (D, st_{\mathcal{A}}) \leftarrow \mathcal{A}(\delta),\\ (com, st_S) \leftarrow Sim(\delta, st_\delta)\end{array}\right]$$

  is negligible in λ. where $\mathcal{O}_P$ and $\mathcal{O}_S$ are defined as follows:
  $\mathcal{O}_P$*: On input a Boolean circuit f, $\mathcal{O}_P$ outputs $\pi \leftarrow \mathsf{Prove}(\delta, com, \tau, f, D(f))$.*
  $\mathcal{O}_S$*: On input a Boolean circuit f, $\mathcal{O}_S$ outputs $\pi \leftarrow Sim(st_S, f, D(f))$.*

5.2 Circuit Transformation

In this subsection, we show how to transform a Boolean circuit query into a set-operation query.

Roughly, we construct a deterministic algorithm that on input a Boolean circuit f output a combined operation $\mathcal{Q}$ (a 'circuit' of unions, intersections and set-differences) such that, for any database D, the set of keys belonging to the output database of querying Boolean circuit f, (i.e., $\{x|\exists v, (x, v) \in D \wedge f(x, v) = 1\}$) equals to the output of combined operation $\mathcal{Q}$ on the corresponding related sets $\{S_i^b\}_{b\in[0,1], i\in[n]}$, which are defined as follows:

$$S_i^b = \{x \in Sup(D)| \text{ the } i\text{-th bit of "}x||v\text{" is } b\}.$$

The construction of above deterministic algorithm is as follows.
Algorithm $\mathcal{Q} \leftarrow \mathsf{Tran}(f)$: On input the boolean circuit $f : \{0,1\}^n \rightarrow \{0,1\}$, $\mathsf{Tran}(f)$ outputs $\mathcal{Q}$, a combined operation having an input of $2n$ sets $(S_1^0, S_1^1, \cdots, S_n^0, S_n^1)$, and outputs a set S'. Here, $\mathcal{Q}$ is constructed as follows:

Tran first associates the i-th input wires of f with the two sets (S_i^0, S_i^1). Supposing that f contains l gates $(n_1, \cdots, n_l)$, without loss of generality, we require the input wires of n_i to be either the input wires of f or the output wires of gates $(n_1, \cdots, n_{i-1})$, and the output of gate n_l is also the output of f. Then for i from 1 to l, we have the following:

1. If gate n_i is "AND" gate, and the sets associated with the two input wires are $(S_{input1}^0, S_{input1}^1), (S_{input2}^0, S_{input2}^1)$, then denote the sets associated with the output wire as $(S_{input1}^1 \cap S_{input2}^1, S_{input1}^0 \cup S_{input2}^0)$.
2. If gate n_i is "OR" gate, and the sets associated with the two input wires are $(S_{input1}^0, S_{input1}^1), (S_{input2}^0, S_{input2}^1)$, then denote the sets associated with the output wire as $(S_{input1}^1 \cup S_{input2}^1, S_{input1}^0 \cap S_{input2}^0)$.
3. If gate n_i is "NOT" gate, and the sets associated with the two input wires are $(S_{input}^0, S_{input}^1)$, then denote the sets associated with the output wire as $(S_{input}^1, S_{input}^0)$.

Supposing that (S^0, S^1) are the sets associated with the output wire of gate n_l, $\mathcal{Q}$ outputs S^1.

Denote by Sup the algorithm that on input a key-value database $D = \{(x_1, v_1), \cdots, (x_m, v_m)\}$, outputs the set of keys belonging to D, i.e., $Sup(D) = \{x_1, \cdots, x_m\}$. We then have the following:

Lemma 11. Tran *is a deterministic algorithm satisfying that for any Boolean circuit f and any key-value databases D,*

$$\mathcal{Q}(S_1^0, S_1^1, \cdots, S_n^0, S_n^1) = Sup(D(f))$$

where $S_i^b = \{x \in Sup(D)|$ the i-th bit of "$x||v$" is $b\}$, $\mathcal{Q} = \mathsf{Tran}(f)$ and $D(f) = \{(x,v) \in D | f(x,v) = 1\}$.

proof sketch. Suppose S_i^0, S_i^1 are the sets associated to the i-th wire defined as in Tran (remind that for input wire $i \in [2n]$, $S_i^b = \{x \in Sup(D)|$ the i-th bit of "$x||v$" is $b\}$), then one can conclude the correctness of above lemma by recursively checking that for each wire i in f, $S_i^b = \{x | \exists v \ s.t. \ (x,v) \in D \wedge$ the value of the i-th wire of $f(x,v)$ is $b\}$, which means that $\mathcal{Q}(S_1^0, S_1^1, \cdots, S_n^0, S_n^1) = \{x | \exists v \ s.t. \ (x,v) \in D \wedge$the value of the output wire of $f(x,v)$ is $1\} = Sup(D(f))$.

5.3 Construction

In this section, we present a construction of the ZK-FEDB from a standard ZK-EDB $(\mathsf{Setup}_D, \mathsf{Com}_D, \mathsf{Prove}_D, \mathsf{Verify}_D)$ and a ZKS with set-operation queries $(\mathsf{Setup}_S, \mathsf{Com}_S, \mathsf{Prove}_S, \mathsf{Verify}_S, \mathsf{SO.Prove}_S, \mathsf{SO.Verify}_S)$.

The construction of ZK-FEDBs is shown in Fig.12.

Zero-Knowledge Functional Elementary Databases

$\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, Setup generates $\delta_D \leftarrow \mathsf{Setup}_D(1^\lambda)$ and $\delta_S \leftarrow \mathsf{Setup}_S(1^\lambda)$. Output CRS $\delta = (\delta_D, \delta_S)$.

$\mathsf{Commit}(\delta, D)$: On input the database D and $\delta = (\delta_D, \delta_S)$, Commit runs $(C_D, \tau_D) \leftarrow \mathsf{Com}_D(\delta_D, D)$. For each $b \in \{0,1\}$, $i \in [2l]$, denote by $S_i^b = \{x \in \mathit{Sup}(D) |$ the i-th bit of $x||v$ is $b\}$. Commit S_i^b using the ZKS scheme, i.e., $(\mathsf{C}_i^b, \tau_i^b) \leftarrow \mathsf{Com}_S(\delta_S, S_i^b)$. Output the commitment $C = (C_D, \{\mathsf{C}_i^b\}_{b\in\{0,1\}, i\in[2l]})$ and the open information $\tau = (\tau_D, \{\tau_i^b\}_{b\in\{0,1\}, i\in[2l]})$.

$\mathsf{Prove}(\delta, C, \tau, f, D(f))$: Prover transforms Boolean circuit f into a combined operations $\mathcal{Q}$ using the algorithm Tran. Run $\pi_Q \leftarrow \mathsf{SO.Prove}_S(\delta_S, \{\mathsf{C}_i^b\}, \{\tau_i^b\}, \mathcal{Q}, \mathit{Sup}(D(f)))$. For each $x \in \mathit{Sup}(D(f))$, run $\pi_x \leftarrow \mathsf{Prove}_D(\delta_D, C_D, \tau_D, x, D(x))$ and for $b \in \{0,1\}, i \in [2l]$, run $\pi_{x,i}^b \leftarrow \mathsf{Prove}_S(\delta_S, \mathsf{C}_i^b, \tau_i^b, x, S_i^b(x))$. Output $\pi = (\pi_Q, \{\pi_x, \pi_{x,i}^b\}_{x\in \mathit{Sup}(D(f)), b\in\{0,1\}, i\in[2l]})$.

$\mathsf{Verify}(\delta, C, f, D(f), \pi)$: Parse C as $(C_D, \{\mathsf{C}_i^b\}_{b\in\{0,1\}, i\in[2l]})$, parse the input π as $(\pi_Q, \{\pi_x, \pi_{x,i}^b\}_{x\in \mathit{Sup}(D(f)), b\in\{0,1\}, i\in[2l]})$ and run $Q = \mathsf{Tran}(f)$. Output 1 iff. the following checks pass:

1. $\mathsf{SO.Verify}_S(\delta_S, \{\mathsf{C}_i^b\}, \mathcal{Q}, \mathit{Sup}(D(f)), \pi_Q) = 1$.
2. For each $x \in \mathit{Sup}(\mathcal{D}(f))$, $\mathsf{Verify}_D(\delta_D, C_D, x, D(x), \pi_x) = 1$.
3. For each $x \in \mathit{Sup}(\mathcal{D}(f))$ and $i \in [2l]$, $\mathsf{Verify}_S(\delta_S, \mathsf{C}_i^{b_{x,i}}, x, 1, \pi_{x,i}^{b_{x,i}}) = 1$ and $\mathsf{Verify}_S(\delta_S, \mathsf{C}_i^{1-b_{x,i}}, x, \perp, \pi_{x,i}^{1-b_{x,i}}) = 1$, where $b_{x,i}$ is the i-th bit of $x||D(x)$.

Fig. 12: ZK-FEDB

Theorem 4. *The scheme shown in Fig.12 is a zero-knowledge functional elementary database scheme.*

Proof. The **completeness** follows from Lemma.11 directly.

To prove the **function binding** property, we will show that, supposing there exists a PPT adversary $\mathcal{A}$ that on input a random CRS δ, outputs a commitment C and a series of valid query-proof tuples $\{f_i, D_i, \pi_{f_i}\}_{i\in[t]}$ such that $\mathsf{Verify}(\delta, C, f_i, D_i, \pi_{f_i}) = 1$ with noticeable property. Then $D = \cup_{i\in[t]} D_i$ is a database satisfying $D(f_i) = D_i$.

First, we claim that D is indeed a database (that is, for each $x \in \mathit{Sup}(D)$, there is at most one value v satisfying $(x, v) \in D$), otherwise, one can break the soundness of the ZK-EDB scheme $(\mathsf{Setup}_D, \mathsf{Com}_D, \mathsf{Prove}_D, \mathsf{Verify}_D)$.

Second, we claim that for each $i \in [t]$, $D(f_i) = D_i$. Denote by $S_i^b = \{x \in \mathit{Sup}(D) |$ the i-th bit of $x||v$ is $b\}$. From the function binding of ZKS with set-operation queires, we know that there exists sets $S_i'^b$ satisfying the first and third checks of the verifier in each proof, which means the following:

1. $\mathcal{Q}_i(S_1'^0, S_1'^1, \cdots, S_n'^0, S_n'^1) = Sup(D_i)$ where $\mathcal{Q}_i = \mathsf{Tran}(f_i)$.
2. For each $i \in [2l]$ and $x \in Sup(D)$, $x \in S_i'^{b_{x,i}}$ and $x \notin S_i'^{1-b_{x,i}}$ where $b_{x,i}$ is the i-th bit of $x||D(x)$.

From the second property above, we have $S_i^b = S_i'^b \cap Sup(D)$. Now, from the first property, we have that $\mathcal{Q}_i(S_1^0, S_1^1, \cdots, S_{2l}^0, S_{2l}^1) = \mathcal{Q}_i(S_1'^0 \cap Sup(D), S_1'^1 \cap Sup(D), \cdots, S_{2l}'^0 \cap Sup(D), S_{2l}'^1 \cap Sup(D)) = D_i \cap Sup(D) = D_i$. From Lemma.11, we have $D(f_i) = D_i$, which concludes the proof.

The **zero-knowledge** property directly follows the zero-knowledge property of ZK-EDB and ZKS with set-operation queries. □

Performance. As shown in Remark.1, our ZKS scheme supports standard batch technique. In Appendix.E, we additionally present a construction of ZK-EDB, which achieves constant-size batched proofs. When utilizing this ZK-EDB and our ZKS, and using the standard batch technique, the performance of our ZK-FEDB is as follows:

	Prover's work	Verifier's work	Communication
Commit	$O(\ell\vert D\vert)\mathsf{EXT} + O(\vert D\vert)h$	N/A	$O(\ell)\mathbb{G}$
Query	$O(\ell\vert D\vert + \vert D\vert\vert f\vert)\mathsf{EXT}$ $+O(\vert D\vert + \ell + \vert f\vert)h$	$O(\ell + \vert f\vert)\mathsf{EXT}$ $+O(\vert D_{output}\vert + \ell + \vert f\vert)h$	$O(\ell + \vert f\vert)\mathbb{G}$

Table 1: Performance of our ZK-FEDB

where ℓ is the bit length of record, $|D|$ and $|D_{output}|$ denote the size of committed database and output database respectly, $|f|$ is the size of query function (for example, a search query can be expressed as a circuit of l AND gates, while a range query can be expressed as a circuit containing no more than 2ℓ AND/OR gates), $\mathbb{G}$ represents a group element, h denotes hashing to a prime and EXT is a λ-bit exponentiation.

Furthermore, by utilizing Boneh et al.'s PoE protocol to reduce the verifier's computation cost, the proof size is approximately $(28\ell + 122|f|)\mathbb{G}$ and the verify cost is approximately $(24\ell + 131|f|)\mathsf{EXT} + (3\ell + 43|f| + |D_{output}|)h$. We hope our work will stimulate more research in this field and bring more efficient constructions of ZK-FEDB.

Applications. Our construction of the ZK-FEDB can be used to construct a Key Transparency system via [CDGM19]'s paradigm. It is easy to see that our construction also satisfies the append-only property. The resulting Key Transparency system achieves enhanced functionality, which enables clients to query public keys in a more flexible manner.

Acknowledgments. We would like to thank the anonymous reviewers for their valuable suggestions. We are supported by the National Natural Science Foundation of China (Grant No. 62372447 and No. 61932019) and Beijing Natural Science Foundation (Grant No. M22003).

References

[AR20] Shashank Agrawal and Srinivasan Raghuraman. Kvac: Key-value commitments for blockchains and beyond. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020*, LNCS 12493, pages 839–869. Springer, 2020.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to iops and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019*, LNCS 11692, pages 561–586. Springer, 2019.

[BdM93] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *Advances in Cryptology - EUROCRYPT 1993*, LNCS 765, pages 274–285. Springer, 1993.

[BH01] Johannes Buchmann and Safuat Hamdy. A survey on iq cryptography. In Kazimierz Alster, Jerzy Urbanowicz, and Hugh C. Williams, editors, *Public-Key Cryptography and Computational Number Theory: Proceedings of the International Conference organized by the Stefan Banach International Mathematical Center Warsaw, Poland, September 11-15, 2000*, pages 1–16, Berlin, New York, 2001. De Gruyter.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT 1997*, LNCS 1233, pages 480–494. Springer, 1997.

[CDG$^+$22] Brian Chen, Yevgeniy Dodis, Esha Ghosh, Eli Goldin, Balachandar Kesavan, Antonio Marcedone, and Merry Ember Mou. Rotatable zero knowledge sets: Post compromise secure auditable dictionaries with application to key transparency. In *Advances in Cryptology – ASIACRYPT 2022*, pages 547—580, Berlin, Heidelberg, 2022. Springer-Verlag.

[CDGM19] Melissa Chase, Apoorvaa Deshpande, Esha Ghosh, and Harjasleen Malvai. Seemless: Secure end-to-end encrypted messaging with less trust. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security - CCS 2019*, pages 1639—1656, New York, NY, USA, 2019. ACM.

[CDV06] Dario Catalano, Yevgeniy Dodis, and Ivan Visconti. Mercurial commitments: Minimal assumptions and efficient constructions. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography - TCC 2006*, LNCS 3876, pages 120–144. Springer, 2006.

[CF13] Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013*, LNCS 7778, pages 55–72. Springer, 2013.

[CFM08] Dario Catalano, Dario Fiore, and Mariagrazia Messina. Zero-knowledge sets with short proofs. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, LNCS 4965, pages 433–450. Springer, 2008.

[CHKO08] Philippe Camacho, Alejandro Hevia, Marcos A. Kiwi, and Roberto Opazo. Strong accumulators from collision-resistant hashing. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *Information Security, 11th International Conference - ISC 2008*, LNCS 5222, pages 471–486. Springer, 2008.

[CHL$^+$05] Melissa Chase, Alexander Healy, Anna Lysyanskaya, Tal Malkin, and Leonid Reyzin. Mercurial commitments with applications to zero-knowledge sets. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, LNCS 3494, pages 422–439. Springer, 2005.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, LNCS 2442, pages 61–76. Springer, 2002.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski, editor, *Advances in Cryptology - CRYPTO 1997*, LNCS 1294, pages 410–424. Springer, 1997.

[CV12] Melissa Chase and Ivan Visconti. Secure database commitments and universal arguments of quasi knowledge. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, LNCS 7417, pages 236–254. Springer, 2012.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, LNCS 2501, pages 125–142. Springer, 2002.

[DHS15] David Derler, Christian Hanser, and Daniel Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015*, LNCS 9048, pages 127–144. Springer, 2015.

[DK02] Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, LNCS 2332, pages 256–271. Springer, 2002.

[EFG22] Liam Eagen, Dario Fiore, and Ariel Gabizon. cq: Cached quotients for fast lookups. Cryptology ePrint Archive, Paper 2022/1763, 2022. https://eprint.iacr.org/2022/1763.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski, editor, *Advances in Cryptology - CRYPTO 1997*, LNCS 1294, pages 16–30. Springer, 1997.

[GM06] Rosario Gennaro and Silvio Micali. Independent zero-knowledge sets. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006*, LNCS 4052, pages 34–45. Springer, 2006.

[GOP+16] Esha Ghosh, Olga Ohrimenko, Dimitrios Papadopoulos, Roberto Tamassia, and Nikos Triandopoulos. Zero-knowledge accumulators and set algebra. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016*, LNCS 10032, pages 67–100. Springer, 2016.

[GOT15] Esha Ghosh, Olga Ohrimenko, and Roberto Tamassia. Zero-knowledge authenticated order queries and order statistics on a list. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security - ACNS 2015*, LNCS 9092, pages 149–171. Springer, 2015.

[GW20] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Paper 2020/315, 2020.

[HHK+21] Yuncong Hu, Kian Hooshmand, Harika Kalidhindi, Seung Jin Yang, and Raluca Ada Popa. Merkle^2: A low-latency transparency log system. In *42nd IEEE Symposium on Security and Privacy, SP 2021*, pages 285–303. IEEE, 2021.

[Lis05] Moses D. Liskov. Updatable zero-knowledge databases. In Bimal K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, LNCS 3788, pages 174–198. Springer, 2005.

[LNTW19] Benoît Libert, Khoa Nguyen, Benjamin Hong Meng Tan, and Huaxiong Wang. Zero-knowledge elementary databases with more expressive queries. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography - PKC 2019*, LNCS 11442, pages 255–285. Springer, 2019.

[LSY+21] Yannan Li, Willy Susilo, Guomin Yang, Tran Viet Xuan Phuong, Yong Yu, and Dongxi Liu. Concise mercurial subvector commitments: Definitions and constructions. In Joonsang Baek and Sushmita Ruj, editors, *Information Security and Privacy - ACISP 2021*, LNCS 13083, pages 353–371. Springer, 2021.

[LY10] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *Theory of Cryptography - TCC 2010*, LNCS 5978, pages 499–517. Springer, 2010.

[MBB+15] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: Bringing key transparency to end users. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 383–398, Washington, D.C., 2015. USENIX Association.

[MRK03] Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science - FOCS'03*, pages 80–91. IEEE Computer Society, 2003.

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005*, LNCS 3376, pages 275–292. Springer, 2005.

[NZ15] Moni Naor and Asaf Ziv. Primary-secondary-resolver membership proof systems. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - TCC 2015*, LNCS 9015, pages 199–228. Springer, 2015.

[ORS04] Rafail Ostrovsky, Charles Rackoff, and Adam Smith. Efficient consistency proofs for generalized queries on a committed database. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming- ICALP 2004*, LNCS 3142, pages 1041–1053, Berlin, Heidelberg, 2004. Springer.

[PTT11] Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Optimal verification of operations on dynamic sets. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011*, LNCS 6841, pages 91–110. Springer, 2011.

[PX09] Manoj Prabhakaran and Rui Xue. Statistically hiding sets. In Marc Fischlin, editor, *Topics in Cryptology - CT-RSA 2009*, LNCS 5473, pages 100–116. Springer, 2009.

[Str19] Michael Straka. Class groups for cryptographic accumulators, 2019. https://www.michaelstraka.com/posts/classgroups/.

[Tam03] Roberto Tamassia. Authenticated data structures. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003*, LNCS 2832, pages 2–5. Springer, 2003.

[TBP$^+$19] Alin Tomescu, Vivek Bhupatiraju, Dimitrios Papadopoulos, Charalampos Papamanthou, Nikos Triandopoulos, and Srinivas Devadas. Transparency logs via append-only authenticated dictionaries. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019*, pages 1299–1316. ACM, 2019.

[TKPS22] Ioanna Tzialla, Abhiram Kothapalli, Bryan Parno, and Srinath T. V. Setty. Transparency dictionaries with succinct proofs of correct operation. In *29th Annual Network and Distributed System Security Symposium, NDSS 2022*. The Internet Society, 2022.

[XLL07] Rui Xue, Ninghui Li, and Jiangtao Li. A new construction of zero-knowledge sets secure in random oracle model. In *The First International Symposium on Data, Privacy, and E-Commerce - ISDPE 2007)*, pages 332–337, 2007.

[XLL08] Rui Xue, Ninghui Li, and Jiangtao Li. Algebraic construction for zero-knowledge sets. *J. Comput. Sci. Technol.*, 23(2):166–175, 2008.

[ZBK$^+$22] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. Caulk: Lookup arguments in sublinear time. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022*, pages 3121–3134. ACM, 2022.

[ZD23] Xinxuan Zhang and Yi Deng. Zero-knowledge functional elementary databases. Cryptology ePrint Archive, Paper 2023/156, 2023.

[ZGK$^+$22] Arantxa Zapico, Ariel Gabizon, Dmitry Khovratovich, Mary Maller, and Carla Ràfols. Baloo: Nearly optimal lookup arguments. Cryptology ePrint Archive, Paper 2022/1565, 2022.

[Zhu09] Huafei Zhu. Mercurial commitments from general rsa moduli and their applications to zero-knowledge databases/sets. In *2009 Second International Workshop on Computer Science and Engineering*, volume 2, pages 289–292, 2009.

[ZKP17] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. An expressive (zero-knowledge) set accumulator. In *2017 IEEE European Sym-*

posium on Security and Privacy (EuroS&P), pages 158–173, 2017.

Appendix

A The proofs of Lemma.2, Lemma.3 and Lemma.4

Lemma. 2 *For any positive integers a, A and B satisfying that $B > A$, we have that:*

$$\mathsf{Dist}(\{x \xleftarrow{\$} \mathbb{Z}_a\}, \{x \mod a | x \xleftarrow{\$} [A, B]\}) \leq \frac{a}{B-A}$$

where Dist *means the statistical distance between distributions.*

Proof. For any integer $t \in \mathbb{Z}_a$, we have that $\Pr[x = t | x \xleftarrow{\$} \mathbb{Z}_a] = 1/a$ and $\Pr[x \bmod a = t | x \xleftarrow{\$} [A, B]] = \frac{\lfloor (B-A)/a \rfloor}{B-A}$ or $\frac{\lfloor (B-A)/a \rfloor + 1}{B-A} \in [\frac{1}{a} - \frac{1}{B-A}, \frac{1}{a} + \frac{1}{B-A}]$. Therefore we have that

$$\begin{aligned} &\mathsf{Dist}(\{x \xleftarrow{\$} \mathbb{Z}_a\}, \{x \mod a | x \xleftarrow{\$} [A, B]\}) \\ =&\Sigma_{t \in [a]} | \Pr[x = t | x \xleftarrow{\$} \mathbb{Z}_a] - \Pr[x \bmod a = t | x \xleftarrow{\$} [A, B]]| \leq \frac{a}{B-A}. \end{aligned}$$

□

Lemma. 3 *For any integers s_1,s_2 and positive integers a, A, B satisfying that $B > A$, $\gcd(s_1, s_2) = 1$, we have that:*

$$\mathsf{Dist}(\{x \xleftarrow{\$} \mathbb{Z}_a\}, \{xs_1 + ys_2 \mod a | x, y \xleftarrow{\$} [A, B]\}) \leq \frac{3a}{B-A}$$

where Dist *means the statistical distance between distributions.*

Proof. For any $t \in \mathbb{Z}_a$, denote by S_t the set of pairs $(x, y) \in \mathbb{Z}_a^2$ satisfying that $xs_1 + ys_2 \equiv t \mod a$. Due to that $\gcd(s_1, s_2) = 1$, there exists integers b_1, b_2 such that $b_1 s_1 + b_2 s_2 = 1$. Therefore, for each $i \in \mathbb{Z}_a$,

$$(tb_1 + is_2 \mod a)s_1 + (tb_2 - is_1 \mod a)s_2 \equiv t \mod a$$

and thus, for each $i \in \mathbb{Z}_a$, $(tb_1 + is_2 \mod a, tb_2 - is_1 \mod a) \in S_t$.

Further more, for each $i \neq j$ in $\mathbb{Z}_a$, it follows that $(tb_1 + is_2 \mod a, tb_2 - is_1 \mod a) \neq (tb_1 + js_2 \mod a, tb_2 - js_1 \mod a)$ (remind that $\gcd(s_1, s_2) = 1$). Therefore for any $t \in \mathbb{Z}_a$, $|S_t| \geq a$. Recall that $\cup_{t \in \mathbb{Z}_a} S_t \subset \mathbb{Z}_a^2$, it follows that $a^2 \leq \Sigma_{t \in \mathbb{Z}_a} |S_t| \leq |\mathbb{Z}_a^2| = a^2$. Therefore, for any $t \in \mathbb{Z}_a$, $|S_t| = a$.

Now, for any $t \in \mathbb{Z}_a$, it follows that

$$\begin{aligned} &\Pr[xs_1 + ys_2 \mod a = t | x, y \xleftarrow{\$} [A, B]] \\ =&\Sigma_{(x', y') \in S_t} \Pr[x \equiv x' \mod a, y \equiv y' \mod a | x, y \xleftarrow{\$} [A, B]]. \end{aligned}$$

Above probability lies in range $[a(\frac{1}{a}-\frac{1}{B-A})^2, a(\frac{1}{a}+\frac{1}{B-A})^2]$. Therefore, if $B-A>a$, we have that

$$\begin{aligned}&\mathsf{Dist}(\{x \xleftarrow{\$} \mathbb{Z}_a\}, \{xs_1+ys_2 \mod a | x, y \xleftarrow{\$} [A,B]\})\\ =&\Sigma_{t\in\mathbb{Z}_a}|\Pr[x=t|x \xleftarrow{\$} \mathbb{Z}_a]-\Pr[xs_1+ys_2 \mod a = t|x,y \xleftarrow{\$} [A,B]]|\\ \leq&\frac{2a}{B-A}+(\frac{a}{B-A})^2 \leq \frac{3a}{B-A}.\end{aligned}$$

In the other hand, if $B-A \leq a$, it follows that $\mathsf{Dist}(\{x \xleftarrow{\$} \mathbb{Z}_a\}, \{xs_1+ys_2 \mod a | x, y \xleftarrow{\$} [A,B]\}) \leq 1 \leq \frac{3a}{B-A}$. The lemma is conclude. □

Lemma. 4 *For any multiplicative group* $\mathbb{G}$ *and group elements* $\mathsf{g}, \mathsf{h} \in \mathbb{G}$*, if there exists* coprime *integers* a, p *satisfying that* $\mathsf{g}^a = \mathsf{h}^p$*, then one can easily compute* h' *satisfying that* $\mathsf{g} = \mathsf{h}'^p$ *from* a, p, g *and* h.

Proof. Due to that $\gcd(a,p)=1$, we can easily compute integers t_1, t_2 such that $t_1a+t_2p=1$. Thus we have that $\mathsf{g}^a=\mathsf{h}^p \Rightarrow \mathsf{g}^{at_1}=\mathsf{h}^{pt_1} \Rightarrow \mathsf{g}^{at_1+pt_2}=\mathsf{h}^{pt_1}\mathsf{g}^{pt_2} \Rightarrow \mathsf{g}=(\mathsf{h}^{t_1}\mathsf{g}^{t_2})^p$. Then, set $\mathsf{h}'=\mathsf{h}^{t_1}\mathsf{g}^{t_2}$ and we have $\mathsf{g}=\mathsf{h}'^p$. □

B The proof of Lemma.5

Proof. **Completeness** is obvious.

The **weak knowledge soundness** follows that: Given an acceptable proof obtained from the prover, from the argument of knowledge property of PoKE, one can extract $\{s_i\}_{i\in[n]}$ satisfying $\mathsf{z}\mathsf{w}^c = \Pi_{i\in[n]}\mathsf{u}_i^{s_i}$. Rewind to step 2 and repeat with fresh random challenge until obtain another acceptable proof. Again, from the argument of knowledge property of PoKE, one can extract $\{s_i'\}_{i\in[n]}$ satisfying $\mathsf{z}\mathsf{w}^{c'} = \Pi_{i\in[n]}\mathsf{u}_i^{s_i'}$ for different challenge c'. Therefore, we have that $\mathsf{w}^{c-c'} = \Pi_{i\in[n]}\mathsf{u}_i^{s_i-s_i'}$, which concludes the weak knowledge soundness.

The simulator of **honest-verifier statistically zero-knowledge** property is constructed as follows: On input random challenges c, ℓ_i (where ℓ_i is the challenge of the i-th PoKE) and statement $(\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w} = \Pi_{i\in[n]}\mathsf{u}_i^{x_i})$, the simulator Sim chooses $s_i \xleftarrow{\$} [2^{2\lambda}B]$ for each $i \in [n]$ and computes $\hat{\mathsf{u}}_i = \mathsf{u}_i^{s_i}$ and $\mathsf{z} = \Pi_{i\in[n]}\hat{\mathsf{u}}/\mathsf{w}^c$. Then Sim runs $\mathsf{PoKE}(\mathsf{u}_i, \hat{\mathsf{u}}_i; s_i)$ with challenge ℓ_i honestly to conclude the simulation.

Therefore, for any fixed $\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w}, c, \{\ell_i\}_{i\in[n]}$, the distribution of the simulation transcript Tran_{Sim} is $\{((\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w}), \{\mathsf{u}_i^{s_i}\}_{i\in[n]}, c, \mathsf{Tran}_{\mathsf{PoKE}_i})|s \leftarrow [2^{2\lambda}B]\}$, where $\mathsf{Tran}_{\mathsf{PoKE}_i} = (\mathsf{g}^{s_i}, \ell_i, \mathsf{u}_i^{\lfloor s_i/\ell_i \rfloor}, \mathsf{g}^{\lfloor s_i/\ell_i \rfloor}, s_i \mod \ell_i)$. And the distribution of the real world transcript Tran_{Real} is $\{((\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w}), \{\mathsf{u}_i^{s_i}\}_{i\in[n]}, c, \mathsf{Tran}_{\mathsf{PoKE}_i})|r_i \leftarrow [2^{2\lambda}B], s_i = r_i + cx_i\}$, where $\mathsf{Tran}_{\mathsf{PoKE}_i} = (\mathsf{g}^{s_i}, \ell_i, \mathsf{u}_i^{\lfloor s_i/\ell_i \rfloor}, \mathsf{g}^{\lfloor s_i/\ell_i \rfloor}, s_i \mod \ell_i)$. Denote by $\mathcal{F}(\{s_i\}_{i\in[n]}) = ((\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w}), \{\mathsf{u}_i^{s_i}\}_{i\in[n]}, c, (\mathsf{g}^{s_i}, \ell_i, \mathsf{u}_i^{\lfloor s_i/\ell_i \rfloor}, \mathsf{g}^{\lfloor s_i/\ell_i \rfloor}, s_i mod ℓ_i)). We have the following:

$$\mathsf{Tran}_{Sim} = \{\mathcal{F}(\{s_i\}_{i\in[n]})|\forall i \in [n], s_i \leftarrow [2^{2\lambda}B]\}$$
$$\mathsf{Tran}_{Real} = \{\mathcal{F}(\{s_i\}_{i\in[n]})|\forall i \in [n], r_i \leftarrow [2^{2\lambda}B], s_i = r_i + cx_i\}$$

As an important observation, $\mathcal{F}(\{s_i\}_{i\in[n]}) = \mathcal{F}(\{s_i \bmod l_i|\mathbb{G}|\}_{i\in[n]})$. As a result, to prove that $\mathsf{Tran}_{Sim} \overset{s}{\approx} \mathsf{Tran}_{Real}$, we only need to prove that for each i, $\{s_i \bmod l_i|\mathbb{G}| \ |s_i \leftarrow [2^{2\lambda}B]\} \overset{s}{\approx} \{s_i \bmod l_i|\mathbb{G}| \ |r_i \leftarrow [2^{2\lambda}B], s_i = r_i + cx_i\}$, which follows from Lemma.2. □

C The proof of Theorem.2

Proof. **Completeness** is obvious.

Soundness follows that: Suppose there exists an adversary $\mathcal{A}$ breaking the soundness property, which means that, upon inputting a random CRS δ, $\mathcal{A}$ can output $(\mathsf{C}, x, v, v', \pi, \pi')$ such that with a noticeable probability, $v \neq v'$ (without loss of generality, we assume that $v = 1$ and $v' = \perp$) and $\mathsf{Verify}(\delta, \mathsf{C}, x, v, \pi) = \mathsf{Verify}(\delta, \mathsf{C}, x, v', \pi') = 1$.

Let $p = \mathcal{H}_{prime}(x)$. Then, from the weak knowledge soundness of NIZK_{DL}, one can extract (a, t) such that $|t| \leq 2^\lambda$ and $\mathsf{g}^{ap} = \mathsf{C}^t$. From Lemma.4 and $\gcd(p, t) = 1$, one can compute h s.t. $\mathsf{C} = \mathsf{h}^p$. From the weak knowledge soundness of NIZK_{2DL}, one can extract (a', b', t') such that $|t'| \leq 2^\lambda$ and $(\mathsf{g}^p)^{a'}\mathsf{C}^{b'} = \mathsf{g}^{t'}$. We therefore have $(\mathsf{g}^{a'}\mathsf{h}^{b'})^p = \mathsf{g}^{t'}$. Again, from Lemma.4 and $\gcd(p, t') = 1$, one can easily compute h' s.t. $\mathsf{h}'^p = \mathsf{g}$, breaking the strong RSA assumption. Therefore, in the generic group model (where strong RSA assumption holds [DF02]), the weak knowledge soundness concludes.

Zero-knowledge property follows the zero-knowledge property of NIZK_{DL} and NIZK_{2DL}. Upon inputting a random CRS δ, the simulator Sim samples $r \leftarrow [2^\lambda B]$ and outputs the commitment $\mathsf{C} = \mathsf{g}^r$. To simulate the proof, Sim directly runs the simulator of NIZK_{DL} and NIZK_{2DL}. From Lemma.2, the distribution of simulated commitment $\{\mathsf{g}^r|r \leftarrow [2^\lambda B]\}$ is statistically indistinguishable from the uniform distribution over group $\langle \mathsf{g} \rangle$. In addition, for any set $S = \{x_1, \cdots, x_m\}$ and $(p_1, \cdots, p_m) = (\mathcal{H}_{prime}(x_1), \cdots, \mathcal{H}_{prime}(x_m))$, the distribution of honest commitment $\{\mathsf{g}^{r\Pi_{i\in[m]}p_i}|r \leftarrow [2^\lambda B]\}$ is also statistically indistinguishable from the uniform distribution over group $\langle g \rangle$ (note that $\gcd(\Pi_{i\in[m]}p_i, Ord(\mathsf{g})) = 1$, and therefore $\mathsf{g}^{\Pi_{i\in[m]}p_i}$ is still a generator of the group $\langle \mathsf{g} \rangle$). As a result, the distributions of the simulated and honest commitments are statistically indistinguishable. Together with the zero-knowledge property of NIZK_{DL} and NIZK_{2DL}, no PPT adversary can tell the simulator apart from the honest committer and prover, which concludes the proof. □

D The proof of Lemma.9 and Lemma.10

In this section, we prove the security of protocols Union-NIZK and Difference-NIZK.

Proof of Lemma.9:

Completeness directly follows the stucture of ZKS commitment.

The simulator of the **zero-knowledge** property only needs to generate $\mathsf{C}_{J_1} = \mathsf{g}^{r_1}$, $\mathsf{C}_{J_2} = \mathsf{g}^{r_2}$ and $\mathsf{C}_I = \mathsf{g}^{r_3}$ by sampling $r_1, r_2, r_3 \leftarrow [2^\lambda B]$, and generate π_1, π_2, π_3 using the simulator of $\mathsf{NIZK}_{pseudo-DDH}$. Then the zero-knowledge property follows from the fact that the distributions of $\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, \mathsf{C}_I = \mathsf{g}^{r_3}$ generated by the simulator are statistically indistinguishable from those generated by an honest prover and the zero-knowledge property of $\mathsf{NIZK}_{pseudo-DDH}$.

The proof of the **special purpose knowledge soundness** is as follows.

From the weak knowledge soundness of $\mathsf{NIZK}_{pseudo-DDH}$, E_1 can extract $w_1 = (x, y, a_1, a_2, a_3, c_1, c_2, c_3)$ such that $|a_1|, |a_2|, |a_3| \leq 2^{5\lambda} B^2$, $|c_1|, |c_2| \leq 2^{6\lambda} B^2$, $|c_3| \leq 2^{8\lambda} B^3$, and $\mathsf{C}_I^{c_1} = \mathsf{g}^{a_1 x}, \mathsf{C}_{J_1}^{c_2} = \mathsf{g}^{a_2 y}, \mathsf{C}_{S_2}^{c_3} = \mathsf{g}^{a_3 xy}$; $w_2 = (x', y', a_1', a_2', a_3', c_1', c_2', c_3')$ such that $|a_1'|, |a_2'|, |a_3'| \leq 2^{5\lambda} B^2$, $|c_1'|, |c_2'| \leq 2^{6\lambda} B^2$, $|c_3'| \leq 2^{8\lambda} B^3$, and $\mathsf{C}_{S_1}^{c_1'} = \mathsf{g}^{a_1' x'}, \mathsf{C}_{J_1}^{c_2'} = \mathsf{g}^{a_2' y'}, \mathsf{C}_U^{c_3'} = \mathsf{g}^{a_3' x' y'}$; and $w_3 = (x'', y'', a_1'', a_2'', a_3'', c_1'', c_2'', c_3'')$ such that $|a_1''|, |a_2''|, |a_3''| \leq 2^{5\lambda} B^2$, $|c_1''|, |c_2''| \leq 2^{6\lambda} B^2$, $|c_3''| \leq 2^{8\lambda} B^3$, and $\mathsf{C}_{S_2}^{c_1''} = \mathsf{g}^{a_1'' x''}, \mathsf{C}_{J_2}^{c_2''} = \mathsf{g}^{a_2'' y''}, \mathsf{C}_U^{c_3''} = \mathsf{g}^{a_3'' x'' y''}$. Here, E_1 outputs $w = (\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, \mathsf{C}_U, w_1, w_2, w_3)$.

Nextly, we show how E_2 works to conclude the proof:

1. On input w, $\mathsf{g}_a \in \mathbb{G}$ and prime $p \in \mathbb{Z}$ satisfying $p \geq 2^{8\lambda} B^3$ and $\mathsf{C}_U = \mathsf{g}_a^p$, E_2 firstly parses w as $(\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, w_1, w_2, w_3)$ and parses w_2 as $(x', y', a_1', a_2', a_3', c_1', c_2', c_3')$. Since $\mathsf{C}_U^{c_3'} = \mathsf{g}^{a_3' x' y'}$, it follows that $\mathsf{g}_a^{c_3' p} = \mathsf{g}^{a_3' x' y'}$. We claim that $p | a_3' x' y'$, otherwise, from Lemma.4, an attacker could easily find a p-root of g and break the strong RSA assumption. Since p is a prime larger than a_3', it follows that $p | x'$ and/or $p | y'$. If $p | x'$, from $\mathsf{C}_{S_1}^{c_1'} = \mathsf{g}^{a_1' x'}$ and $\gcd(p, c_1') = 1$, E_2 can easily compute g_b by Lemma.4 such that $\mathsf{C}_{S_1} = \mathsf{g}_b^p$. If $p | y'$, E_2 can similarly compute g_c such that $\mathsf{C}_{J_1} = \mathsf{g}_c^p$. Parse w_1 as $(x, y, a_1, a_2, a_3, c_1, c_2, c_3)$, then it follows that $\mathsf{g}_c^{p c_2} = \mathsf{C}_{J_1}^{c_2} = \mathsf{g}^{a_2 y}$. We claim that $p | y$, otherwise, from Lemma.4, an attacker could break the strong RSA assumption. As $\mathsf{C}_{S_2}^{c_3} = \mathsf{g}^{a_3 xy}$ and $\gcd(p, c_3) = 1$, E_2 can easily compute g_d such that $\mathsf{C}_{S_2} = \mathsf{g}_d^p$.
2. On input w and $a, b, p \in \mathbb{Z}$ satisfying that p is a prime larger than $2^{8\lambda} B^3$ and $\mathsf{C}_U^a \cdot \mathsf{g}^{bp} = \mathsf{g}$, E_2 firstly parses w as $(\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, w_1, w_2, w_3)$ and parses w_2 as $(x', y', a_1', a_2', a_3', c_1', c_2', c_3')$. Since $\mathsf{C}_U^{c_3'} = \mathsf{g}^{a_3' x' y'}$, it follows that $\gcd(p, x') = 1$, otherwise, from Lemma.4 and $\mathsf{C}_U^a \cdot \mathsf{g}^{bp} = \mathsf{g}$, an attacker could easily find a p-root of g and break the strong RSA assumption. Then, $\gcd(p, a_1' x') = 1$, and E_2 can easily find integers α, β satisfying $\alpha p + \beta a_1' x' = 1$, and then $\mathsf{C}_{S_1}^{c_1' \beta} \mathsf{g}^{\alpha p} = \mathsf{g}$. Setting $a' = \beta c_1'$ and $b' = \alpha$, then $\mathsf{C}_{S_1}^{a'} \cdot \mathsf{g}^{b' p} = \mathsf{g}$. In the same strategy, E_2 can compute a'', b'' from w_3 such that $\mathsf{C}_{S_2}^{a''} \cdot \mathsf{g}^{b'' p} = \mathsf{g}$, which concludes the proof. □

Proof of Lemma.10:

Completeness directly follows the stucture of ZKS commitment.

The simulator of the **zero-knowledge** property only needs to generate $\mathsf{C}_I = \mathsf{g}^{r_1}$ and $\mathsf{C}_J = \mathsf{g}^{r_2}$ by sampling $r_1, r_2 \leftarrow [2^\lambda B]$, and generate π_1, π_2, π_3 using the simulator of $\mathsf{NIZK}_{pseudo-DDH}$ and $\mathsf{NIZK}_{coprime}$. Then the zero-knowledge property follows from the fact that the distributions of $\mathsf{C}_{J_1}, \mathsf{C}_{J_2}$ generated by the simulator are statistically indistinguishable from those generated by an honest prover and the zero-knowledge property of $\mathsf{NIZK}_{pseudo-DDH}$ and $\mathsf{NIZK}_{coprime}$.

The proof of the **special purpose knowledge soundness** is as follows.

From the weak knowledge soundness of $\mathsf{NIZK}_{pseudo-DDH}$, E_1 can extract $w_1 = (x, y, a_1, a_2, a_3, c_1, c_2, c_3)$ such that $|a_1|, |a_2|, |a_3| \leq 2^{5\lambda}B^2$, $|c_1|, |c_2| \leq 2^{6\lambda}B^2$, $|c_3| \leq 2^{8\lambda}B^3$, and $\mathsf{C}_I^{c_1} = \mathsf{g}^{a_1 x}, \mathsf{C}_D^{c_2} = \mathsf{g}^{a_2 y}, \mathsf{C}_{S_1}^{c_3} = \mathsf{g}^{a_3 xy}$; and $w_2 = (x', y', a_1', a_2', a_3',\ c_1', c_2', c_3')$ such that $|a_1'|, |a_2'|, |a_3'| \leq 2^{5\lambda}B^2$, $|c_1'|, |c_2'| \leq 2^{6\lambda}B^2$, $|c_3'| \leq 2^{8\lambda}B^3$,and $\mathsf{C}_I^{c_1'} = \mathsf{g}^{a_1' x'}, \mathsf{C}_J^{c_2'} = \mathsf{g}^{a_2' y'}, \mathsf{C}_{S_2}^{c_3'} = \mathsf{g}^{a_3' x' y'}$. From the weak knowledge soundness of $\mathsf{NIZK}_{coprime}$, E_1 can extract $w_3 = (t_1, t_2, c)$ such that $c \leq 2^{5\lambda}B^2$ and $\mathsf{C}_D^{t_1}\mathsf{C}_{S_2}^{t_2} = \mathsf{g}^c$. Here, E_1 outputs $w = (\mathsf{C}_D, \mathsf{C}_{S_2}, w_1, w_2, w_3)$.

Nextly, we show how E_2 works to conclude the proof:

1. On input w, $\mathsf{g}_a \in \mathbb{G}$ and prime $p \in \mathbb{Z}$ satisfying $p \geq 2^{8\lambda}B^3$ and $\mathsf{C}_D = \mathsf{g}_a^p$, E_2 firstly parses w as $(\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, w_1, w_2, w_3)$ and parses w_1 as $(x, y, a_1, a_2, a_3, c_1, c_2, c_3)$. Since $\mathsf{C}_D^{c_2} = \mathsf{g}^{a_2 y}$, it follows that $\mathsf{g}_a^{c_2 p} = \mathsf{g}^{a_2 y}$. We claim that $p | a_2 y$, otherwise, from Lemma.4, an attacker could easily find a p-root of g and break the strong RSA assumption. Since p is a prime larger than a_2, it follows that $p|y$ and $\mathsf{C}_{S_1}^{c_3} = \mathsf{g}^{a_3 xy} = (\mathsf{g}^{a_3 xy/p})^p$. As $\gcd(p, c_3) = 1$, E_2 can easily compute g_b by Lemma.4 such that $\mathsf{C}_{S_1} = \mathsf{g}_b^p$. Meanwhile, parse w_3 as (t_1, t_2, c), it follows that $\mathsf{g}_a^{pt_1}\mathsf{C}_{S_2}^{t_2} = \mathsf{C}_D^{t_1}\mathsf{C}_{S_2}^{t_2} = \mathsf{g}^c$. Since $\mathsf{C}_{S_2}^{c_3'} = \mathsf{g}^{a_3' x' y'}$, we have $\gcd(p, a_3' x' y') = 1$, otherwise, from Lemma.4 and $\mathsf{g}_a^{pt_1}\mathsf{C}_{S_2}^{t_2} = \mathsf{g}^c$, an attacker could easily break the strong RSA assumption. Then, E_2 can efficiently find $\alpha, \beta \in \mathbb{Z}$ such that $\alpha p + \beta a_3' x' y' = 1$. By setting $a' = \beta c_3', b' = \alpha$, E_2 outputs a', b' such that $\mathsf{C}_{S_2}^{a'}\mathsf{g}^{b' p} = \mathsf{g}$.
2. On input w and $a, b, p \in \mathbb{Z}$ satisfying that p is a prime larger than $2^{8\lambda}B^3$ and $\mathsf{C}_D^a \cdot \mathsf{g}^{bp} = \mathsf{g}$, E_2 firstly parses w as $(\mathsf{C}_{J_1}, \mathsf{C}_{J_2}, w_1, w_2, w_3)$ and parses w_1 as $(x, y, a_1, a_2, a_3, c_1, c_2, c_3)$. Since $\mathsf{C}_D^{c_2} = \mathsf{g}^{a_2 y}$, it follows that $\gcd(p, y) = 1$, otherwise, from Lemma.4 and $\mathsf{C}_D^a \cdot \mathsf{g}^{bp} = \mathsf{g}$, an attacker could easily find a p-root of g and break the strong RSA assumption. There are two cases, $p \nmid x$ or $p|x$. When $p \nmid x$, it follows that $\gcd(p, a_3 xy) = 1$. E_2 can efficiently compute $\alpha, \beta \in \mathbb{Z}$ such that $\alpha p + \beta a_3 xy$. By setting $a' = \beta c_3', b' = \alpha$, E_2 outputs a', b' such that $\mathsf{C}_{S_1}^{\beta c_3}\mathsf{g}^{\alpha p} = \mathsf{g}$. When $p|x$, it follows that $\mathsf{C}_I^{c_1} = \mathsf{g}^{a_1 x} = (\mathsf{g}^{a_1 x/p})^p$. From Lemma.4, E_2 can compute g_a such that $\mathsf{C}_I = \mathsf{g}_a^p$. Parse w_2 as $(x', y', a_1', a_2', a_3', c_1', c_2', c_3')$, it follows that $\mathsf{g}_a^{c_1' p} = \mathsf{C}_I^{c_1'} = \mathsf{g}^{a_1' x'}$. Thus, $p|x'$, otherwise an attacker could easily break the strong RSA assumption. Then one have that $\mathsf{C}_{S_2}^{c_3'} = \mathsf{g}^{a_3' x' y'} = (\mathsf{g}^{a_3' x'_p y'})^p$, and therefore, from Lemma.4, E_2 can efficiently compute g_b such that $\mathsf{C}_{S_2} = \mathsf{g}_b^p$. □

E Constant-Size Zero-Knowledge Elementary Databases

In this section, we present a standard ZK-EDB scheme achieving constant commitment and proof size from groups of unknown orders.

Standard zero-knowledge elementary database scheme

$\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, Setup generates the description of an unknown-order group $\mathbb{G} \leftarrow GGen(\lambda)$ and a random group element $\mathsf{g} \xleftarrow{\$} \mathbb{G}$. Suppose B is the upper bound of $\mathbb{G}$ (i.e., $B \geq |\mathbb{G}|$). Sample the description of a hash function $\mathcal{H}_{prime}$ that on input a string, outputs a random prime larger than $2^{\lambda+\ell}B$. Output CRS $\delta = (\mathbb{G}, \mathsf{g}, B, \mathcal{H}_{prime})$.

$\mathsf{Commit}(\delta, D)$: On input the set $D = \{(x_i, v_i)\}_{i\in[m]}$ where for each $i \in [m]$, $(x_i, v_i) \in \{0,1\}^l \times \{0,1\}^l$, Commit hashes keys into large primes, i.e., for each $i \in [m]$, $p_i \leftarrow \mathcal{H}_{prime}(x_i)$. Denote by $t = \Sigma_{i\in[m]} v_i \cdot \Pi_{j\neq i} p_j \cdot (\Pi_{j\neq i} p_j^{-1} \mod p_i)$ such that $t \mod p_i = v_i$. Sample $r, r' \leftarrow [2^\lambda B]$ uniformly, and output the commitment $C = (\mathsf{c}_1, \mathsf{c}_2) = (\mathsf{g}^{r\Pi_{i\in[m]}p_i}, \mathsf{g}^{t+r'\Pi_{i\in[m]}p_i})$ and the open information $\tau = (r, r', p_1, \cdots, p_m, D)$.

$\mathsf{Prove}(\delta, (C, \tau), x, D(x))$: Parse the input τ as $(r, r', p_1, \cdots, p_m, D)$.

- If $D(x) = v \neq \perp$, it follows that $p = \mathcal{H}_{prime}(x) \in \{p_1, \cdots, p_m\}$. Prove runs $\pi_1 \leftarrow \mathsf{NIZK}_{DL}(\mathsf{g}^p, \mathsf{c}_1; r\Pi_{i\in[m]}p_i/p)$, $\pi_2 \leftarrow \mathsf{NIZK}_{DL}(\mathsf{g}^p, \mathsf{c}_2 \cdot \mathsf{g}^{-v}; (t + r\Pi_{i\in[m]}p_i - v)/p)$ and outputs $\pi = (\pi_1, \pi_2)$.
- If $D(x) = \perp$, which means that $p = \mathcal{H}_{prime}(x) \notin \{p_1, \cdots, p_m\}$ and $\gcd(p, r\Pi_{i\in[m]}p_i) = 1$, Prove finds a, b such that $ap + br\Pi_{i\in[m]}p_i = 1$, and outputs $\pi \leftarrow \mathsf{NIZK}_{2DL}(\mathsf{g}^p, \mathsf{c}_1, \mathsf{g}; a, b)$.

$\mathsf{Verify}(\delta, C, x, D(x), \pi)$: If $D(x) = v \neq \perp$, check whether π consists of two valid NIZK_{DL} proofs for statement $(\mathsf{g}^p, \mathsf{c}_1)$ and $(\mathsf{g}^p, \mathsf{c}_2 \cdot \mathsf{g}^{-v})$. If $D(x) = \perp$, check whether π is a valid NIZK_{2DL} proof for statement $(\mathsf{g}^p, \mathsf{c}_2, \mathsf{g}) \in \mathcal{R}_{2DL}$. Output 1 iff. the check pass.

Fig. 13: Protocol ZK-EDB

Theorem 5. *The protocol constructed in Fig.13 is a secure ZK-EDB scheme in the generic group model and random oracle model.*

proof sketch. The proof of the above theorem is similar to that of Theorem.2. The **completeness** is oblivious and the zero-knowledge property follows the zero-knowledge property of NIZK_{DL} and NIZK_{2DL} and the fact that the distribution of c_1, c_2 is statistically indistinguishable from the uniform distribution over group $\langle g \rangle$.

Because c_1 is actually a ZKS for set $Sup(D)$, from the soundness of ZKS scheme, no adversary can prove that $x \in Sup(D)$ and $x \notin Sup(D)$ simultaneously. Now, suppose there exists an adversary that can simultaneously prove $(x, v) \in D$ and $(x, v') \in D$, and $v \neq v'$. It then means that the adversary can generate two valid proofs NIZK_{DL} for statements $(\mathsf{g}^p, \mathsf{c}_2 \cdot \mathsf{g}^{-v}), (\mathsf{g}^p, \mathsf{c}_2 \cdot \mathsf{g}^{-v'})$, where $p = \mathcal{H}_{prime}(x)$. From the weak knowledge soundness of NIZK_{DL}, one can extract $(a, b), (a', b')$ such that $b, b' \leq 2^{\lambda}$, $\mathsf{g}^{pa} = (\mathsf{c}_2 \cdot \mathsf{g}^{-v})^b$ and $\mathsf{g}^{pa'} = (\mathsf{c}_2 \cdot \mathsf{g}^{-v'})^{b'}$. We then have $(\mathsf{g}^{ab'-a'b})^p = \mathsf{g}^{(v'-v)bb'}$. Furthermore, since $\gcd(p, (v'-v)bb') = 1$, an attacker could easily find a p-root of g and break the strong RSA assumption, which concludes the proof.

Remark 3. One can use the batch technique put forward in [BBF19] to batch the (non-)membership proofs. For example, to prove that $(x'_1, v'_1), \cdots, (x'_t, v'_t) \in D$, the prover hashes the keys into primes $p'_1, \cdots p'_t$ by $\mathcal{H}_{prime}$ and generates the proof $\pi = (\pi_1, \pi_2)$, $\pi_1 \leftarrow \mathsf{NIZK}_{DL}(\mathsf{g}^{\mathit{\Pi}_{i\in[t]}p'_i}, \mathsf{C}; r\mathit{\Pi}_{i\in[m]}p_i/\mathit{\Pi}_{i\in[t]}p'_i)$, $\pi_2 \leftarrow \mathsf{NIZK}_{DL}(\mathsf{g}^p, \mathsf{c}_2 \cdot \mathsf{g}^{-\tilde{v}}; (t + r\mathit{\Pi}_{i\in[m]}p_i - \tilde{v})/\mathit{\Pi}_{i\in[t]}p'_i))$ where $\tilde{v}$ is the least integer that satisfies $\tilde{v} \equiv v'_i \mod \mathit{\Pi}_{i\in[t]}p'_i)$ for each $i \in [t]$. To prove that $x'_1, \cdots, x'_t \notin Sup(D)$, the prover hashes them into primes $p'_1, \cdots p'_t$ by $\mathcal{H}_{prime}$ and finds integers a, b such that $a\mathit{\Pi}_{i\in[t]}p'_i + br\mathit{\Pi}_{i\in[m]}p_i = 1$, and then outputs $\pi \leftarrow \mathsf{NIZK}_{2DL}(\mathsf{g}^{\mathit{\Pi}_{i\in[t]}p'_i}, \mathsf{C}, \mathsf{g}; a, b)$.