Zero-Knowledge Functional Elementary Databases

Xinxuan Zhang   Yi Deng

December 5, 2023

State Key Laboratory of Information Security, Institute of Information Engineering, CAS

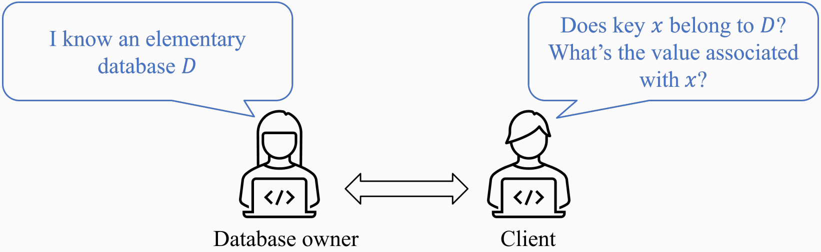
School of Cyber Security, University of Chinese Academy of Sciences

Backgroud

Zero-Knowledge Elementary Databases

Consider the following scenario:

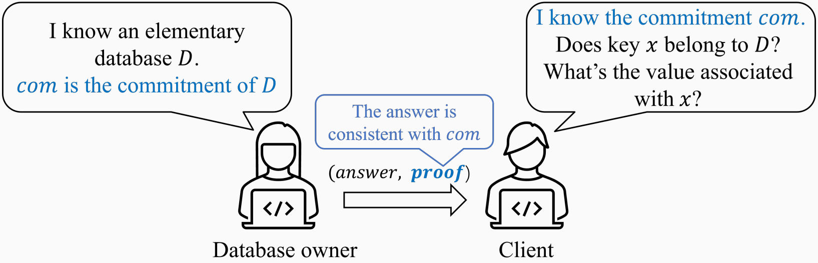
Let $D = \{(x, v)\}$ be an elementary database $((x, v) \in D, (x, v') \in D \Rightarrow v = v')$.



- The database owner cannot answer the queries inconsistently.
- The client cannot learn extra knowledge.

Zero-Knowledge Elementary Databases

Zero-Knowledge Elementary Databases (ZK-EDB):

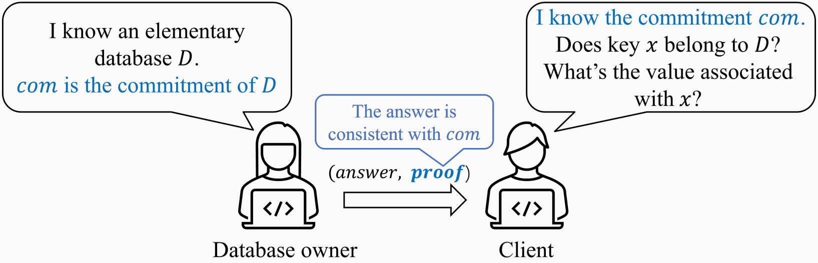


A ZK-EDB consists of four algorithms (Setup, Com, Prove, Verify):

- Soundness: The database owner/committer cannot answer the same queries inconsistently.
- Zero-knowledge: The commitment and proof will not reveal any extra knowledge, **including the size of** D. The size of D is not contained in the input of simulator.

Zero-Knowledge Elementary Databases

Zero-Knowledge Elementary Databases (ZK-EDB):



Application: End-to-end encrypted communication (E2EE) systems

Provide an auditable and queryable directory of their users' public keys (Key Transparent system).

The Quries of ZK-EDB

Most constructions:

- Follow the paradigm of Chase et al.
- Only support membership queries.

Libert et al.'s zero-knowledge expressive elementary databases:

- Modify Chase et al.'s paradigm.
- Support range queries over keys and/or values.

Question:
Can we construct ZK-EDB supporting richer queries?

Difficulties

A naive attempt:

Commitment + zk-SNARKs

Difficulties

A naive attempt:

Commitment + zk-SNARKs

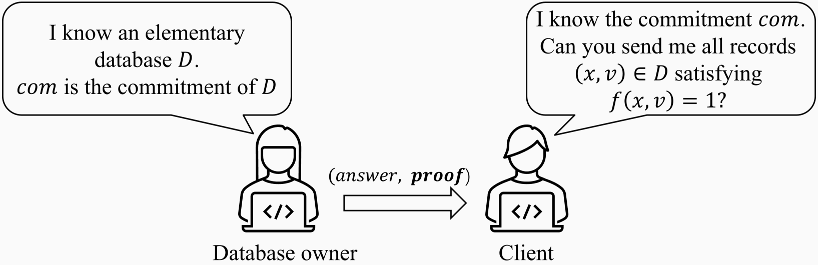
However, this attempt would fail due to the potential revelation of the database size.

- Almost all zk-SNARKs expose the length of the witness.
- For generalize functional query, the witness must include all records in database to ensure the correctness of query.

Our Contributions

Our Contributions

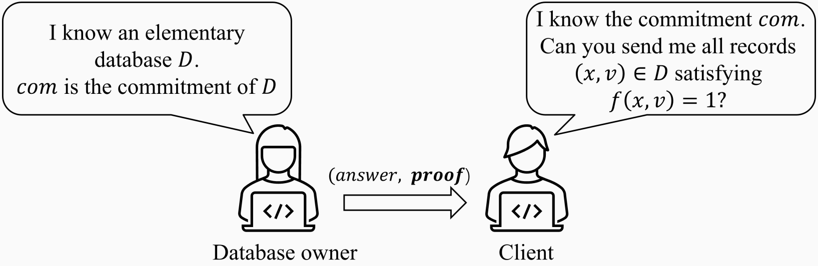
Zero-Knowledge Functional Elementary Databases (ZK-FEDB)



- Allow **the most generalize functional queries**: For any Boolean circuit f, clients can query that: "Send me all records $(x, v) \in D$ satisfying $f(x, v) = 1$."
- Function Binding (Soundness) and Zero-Knowledge.

Our Contributions

Zero-Knowledge Functional Elementary Databases (ZK-FEDB)



Construction based on unknown-order group.

- Proof size: $O(|(x, v)| + |f|)$ (independent of $|D|$)
- Secure in the random oracle model and generic group model.

Technique Contributions

Our technical constribution is two-fold.

- **A new variant of zero-knowledge sets (ZKS)**: Support combined operations queries on committed sets.
- **A new transformation technique**: Transform the query of Boolean circuit into a query of combined operations on related sets.

Note.
ZKS: the "set" version of ZK-EDB, committing sets rather than databases.
Combined operation: a "circuit" with gates "intersection", "union" and "set-difference".

Zero-Knowledge Sets with Set-Operation Queries

Start from RSA Accumulators

RSA Accumulator

- g: The ganerator of an unknown-order group.
- Commitment of set $S = \{x_i\}_{i \in [m]}$:

$$\mathsf{C} = \mathsf{g}^{\prod_{i \in [m]} p_i}$$

  where $p_i = \mathcal{H}_{prime}(x_i)$ is a prime.
- Membership proof of $x_j \in S$: g_j satisfying $\mathsf{g}_j^{p_j} = \mathsf{C}$.
- Non-membership proof of $x \notin S$: (a, b) satisfying $\mathsf{C}^a \mathsf{g}^{b\mathcal{H}_{prime}(x)} = \mathsf{g}$.

A pair of membership proof and non-membership proof of same element can be used to break strong RSA assumption.

Basic Set Operations

Basic Set Relation:
"Intersection, Union, Set-Defference"

$\Downarrow$

Simpler set relations:

- Disjoint relation
$$\{(J_0, J_1) | J_0 \cap J_1 = \emptyset\}$$
- Union among disjoint relation
$$\left\{ (U, J_0, J_1) \middle| \begin{array}{l} U = J_0 \cup J_1 \wedge \\ J_0 \cap J_1 = \emptyset \end{array} \right\}$$
.

Basic Set Relation on Commitments:
"Intersection, Union, Set-Defference"

$\Downarrow$

Group Element Relations:

- Co-prime relation
$$\left\{ (\mathrm{C}_1, \mathrm{C}_2) \middle| \begin{array}{r} \exists a, b \in \mathbb{Z} \; s.t. \\ \gcd(a, b) = 1 \wedge \\ (\mathrm{C}_1, \mathrm{C}_2) = (\mathrm{g}^a, \mathrm{g}^b) \end{array} \right\}.$$
- DDH tuples relation

Zero-Knowledge Sets

RSA accumulators can be convert into ZKS by adding randomness r to provide privacy.

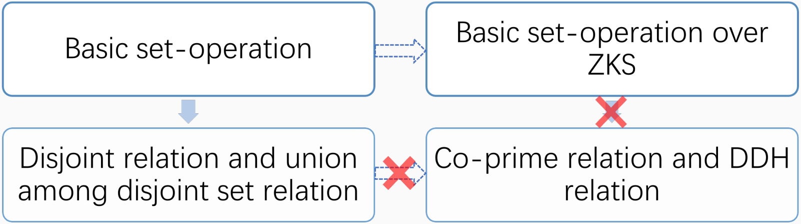
$$\mathrm{g}^{\prod_{i\in[m]} p_i} \Rightarrow \mathrm{g}^{r\cdot\prod_{i\in[m]} p_i}.$$

Zero-Knowledge Sets

RSA accumulators can be convert into ZKS by adding randomness r to provide privacy.

$$g^{\prod_{i\in[m]} p_i} \Rightarrow g^{r\cdot\prod_{i\in[m]} p_i}.$$

Question:



Zero-Knowledge Sets

Key observation:
In ZKS commitment, randomness is sampled from small and bounded range of $[0, B]$.

- Let A, B be disjoint sets, $g^{r \cdot \mathcal{H}_{prime}(A)}, g^{r' \cdot \mathcal{H}_{prime}(B)}$ are their ZKS commitments.

$$\gcd(r \cdot \mathcal{H}_{prime}(A), r' \cdot \mathcal{H}_{prime}(B)) = \gcd(r, r') \text{ is small}$$

- Let A, B be disjoint sets, $U = A \cup B$, $g^{r \cdot \mathcal{H}_{prime}(A)}, g^{r' \cdot \mathcal{H}_{prime}(B)}$, $g^{r'' \cdot \mathcal{H}_{prime}(U)}$ are their ZKS commitments.

$$(g^{r \cdot \mathcal{H}_{prime}(A)}, g^{r' \cdot \mathcal{H}_{prime}(B)}, g^{r'' \cdot \mathcal{H}_{prime}(U)}) \text{ is close to a DDH-tuple}$$

We call above two relations as pseudo-coprime relation and pseudo-DDH relation.

Zero-Knowledge Protocol

Tools:

1. Schnorr's Σ-protocol for bounded discrete-log:

$$\mathcal{R}_{boundedDL} = \{(\mathsf{u}, \mathsf{w}, T; x) | \mathsf{u}^x = \mathsf{w} \wedge |x| \leq T\}$$

2. (A new variant of) Boneh et al.'s ZK-argument for multidimensional discrete-log.

$$\mathcal{R}_{multiDL} = \{(\{\mathsf{u}_i\}_{i\in[n]}, \mathsf{w}; \{x_i\}_{i\in[n]}) | \Pi_{i\in[n]} \mathsf{u}_i^{x_i} = \mathsf{w}\}$$

Note: Both of above protocols only achieve a weak soundness due to that "Computing $\mathsf{g}^{\frac{1}{a}}$ in an unknown-order group is hard". Luckily, it is sufficient for our construction.

Zero-Knowledge Protocol for Pseudo-Coprime Relation

Prove pseudo-coprime relation:

$$\{(u, v, T; a_1, a_2) | u = g^{a_1}, v = g^{a_2}, \gcd(a_1, a_2) < T\}$$

⇕ Equivalence

Find a, b and prove:

$$u^a \cdot v^b = g^{\gcd(a_1,a_2)} \wedge \gcd(a_1, a_2) < T$$

⇓ Add randomness $r < B$ to mask $g^{\gcd(a_1,a_2)}$.

ZK protocol

Send $g^{r\cdot\gcd(a_1,a_2)}$ and prove:

$$u^{ra} \cdot v^{rb} = g^{r\cdot\gcd(a_1,a_2)} \wedge r \cdot \gcd(a_1, a_2) < BT$$

- $u^{ra} \cdot v^{rb} = g^{r\cdot\gcd(a_1,a_2)}$: ZK for multi-DL
- $r \cdot \gcd(a_1, a_2) < BT$: Schnorr's Σ-protocol

- Only achieve a weak soundness. (The GCD of exponents might be larger than T, however, it is still bounded by a proper upper bound.)
- One can use the Fiat-Shamir heuristic to obtain the non-interactive version.

Zero-Knowledge Protocol for Pseudo-DDH Relation

Prove pseudo-DDH relation:
$u = g^{a_1 x}, v = g^{a_2 y}, w = g^{a_3 xy} \wedge a_1, a_2, a_3 < T$

Choose randomness $r_1, r_2 < B$ to generate a close DDH-tuple: $(u' = g^{r_1 x}, v' = g^{r_2 y},\ w' = g^{r_1 r_2 xy})$

ZK protocol

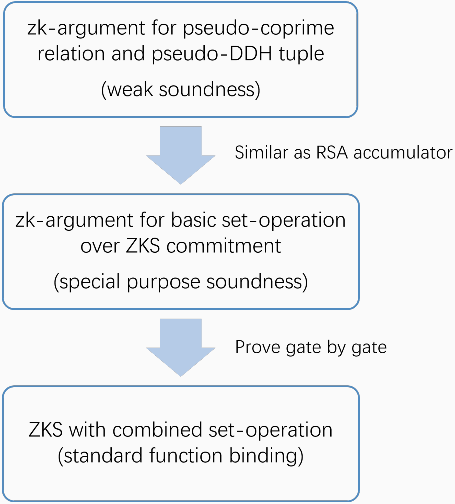
Send (u', v', w') and prove:

(u', v', w') is a DDH-tuple $\wedge$ (u', v', w') and (u, v, w) is close

Boneh et al's PoDDH — Schnorr's Σ-protocol

- Only achieve a weak soundness. (That is, the statement might not close to DDH-tuple as we required, however, it is still close enough.)
- One can use the Fiat-Shamir heuristic to obtain the non-interactive version.

Zero-Knowledge Sets



From Boolean Circuit Queries to Set-Operation

From Boolean Circuit Queries to Set-Operation

Our goal:

Query of Boolean circuit f over a set S
(requesting $S_{output} := \{x | x \in S \wedge f(x) = 1\}$)

$$\Downarrow$$

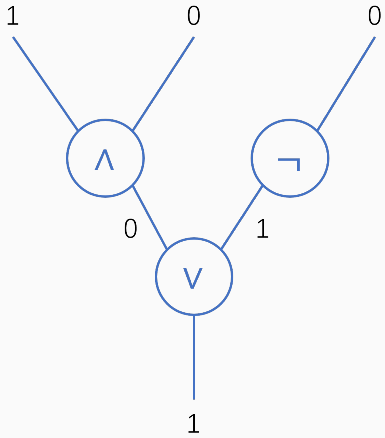
Query of combined operations $\mathcal{Q}$ on related sets
$S_i^b := \{x | x \in S \wedge \text{ the i-th bit of ``}x\text{'' is } b\}$
(requesting $S_{output} := \mathcal{Q}(\{S_i^b\})$)

From Boolean Circuit Queries to Set-Operation

Let f be a Boolean circuit.

- When running f on an input, each wire in f has a value.

Example:



From Boolean Circuit Queries to Set-Operation

Let f be a Boolean circuit.

- When running f on an input, each wire in f has a value.
- When running f on a set S, according the value of wire, each wire i can be associated with two subsets $\{S_i^b\}_{b\in\{0,1\}}$. That is, $S_i^b := \{x | x \in S \wedge \text{ the value of i-th wire of } f(x) \text{ is } b\}$

From Boolean Circuit Queries to Set-Operation

Let f be a Boolean circuit.

- When running f on an input, each wire in f has a value.
- When running f on each element of a set S, according the value of wire, each wire i can be associated with two sets $\{S_i^b\}_{b\in\{0,1\}}$. That is, $S_i^b := \{x | x \in S \wedge \text{ the value of i-th wire of } f(x) \text{ is } b\}$

Key Observation:

- For each input wire i, $S_i^b = \{x | x \in S \wedge \text{ the i-th bit of ``}x\text{'' is } b\}$.

From Boolean Circuit Queries to Set-Operation

Let f be a Boolean circuit.

- When running f on an input, each wire in f has a value.
- When running f on each element of a set S, according the value of wire, each wire i can be associated with two sets $\{S_i^b\}_{b\in\{0,1\}}$. That is, $S_i^b := \{x | x \in S \wedge \text{ the value of i-th wire of } f(x) \text{ is } b\}$

Key Observation:

- For each input wire i, $S_i^b = \{x | x \in S \wedge \text{ the i-th bit of "}x\text{" is } b\}$.
- For the output wire *output*, the second associated set S_{output}^1, is exactly the answer of the query of Boolean circuit f.

From Boolean Circuit Queries to Set-Operation

Let f be a Boolean circuit.

- When running f on an input, each wire in f has a value.
- When running f on each element of a set S, according the value of wire, each wire i can be associated with two sets $\{S_i^b\}_{b \in \{0,1\}}$. That is, $S_i^b := \{x | x \in S \wedge \text{ the value of i-th wire of } f(x) \text{ is } b\}$

Key Observation:

- For each input wire i, $S_i^b = \{x | x \in S \wedge \text{ the i-th bit of ``}x\text{'' is } b\}$.
- For the output wire *output*, the second associated set S^1_{output}, is exactly the answer of the query of Boolean circuit f.
- For any AND gate in f with input wires a, b and output wire c, $S_c^0 = S_a^0 \cup S_b^0$ and $S_c^1 = S_a^1 \cap S_b^1$.

From Boolean Circuit Queries to Set-Operation

Let f be a Boolean circuit.

- When running f on an input, each wire in f has a value.
- When running f on each element of a set S, according the value of wire, each wire i can be associated with two sets $\{S_i^b\}_{b\in\{0,1\}}$. That is, $S_i^b := \{x|x \in S \wedge \text{ the value of i-th wire of } f(x) \text{ is } b\}$

Key Observation:

- For each input wire i, $S_i^b = \{x|x \in S \wedge \text{ the i-th bit of "}x\text{" is } b\}$.
- For the output wire *output*, the second associated set S^1_{output}, is exactly the answer of the query of Boolean circuit f.
- For any AND gate in f with input wires a, b and output wire c, $S_c^0 = S_a^0 \cup S_b^0$ and $S_c^1 = S_a^1 \cap S_b^1$.
- For any OR gate with input wires a, b and output wire c, $S_c^0 = S_a^0 \cap S_b^0$ and $S_c^1 = S_a^1 \cup S_b^1$.

From Boolean Circuit Queries to Set-Operation

Let f be a Boolean circuit.

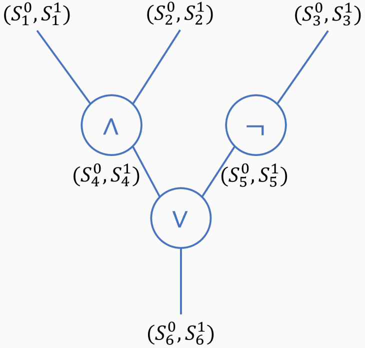
- When running f on an input, each wire in f has a value.
- When running f on each element of a set S, according the value of wire, each wire i can be associated with two sets $\{S_i^b\}_{b\in\{0,1\}}$. That is, $S_i^b := \{x|x \in S \wedge \text{ the value of i-th wire of } f(x) \text{ is } b\}$

Key Observation:

- For each input wire i, $S_i^b = \{x|x \in S \wedge \text{ the i-th bit of "}x\text{" is } b\}$.
- For the output wire *output*, the second associated set S_{output}^1, is exactly the answer of the query of Boolean circuit f.
- For any AND gate in f with input wires a, b and output wire c, $S_c^0 = S_a^0 \cup S_b^0$ and $S_c^1 = S_a^1 \cap S_b^1$.
- For any OR gate with input wires a, b and output wire c, $S_c^0 = S_a^0 \cap S_b^0$ and $S_c^1 = S_a^1 \cup S_b^1$.
- For any NOT gate with input wire a and output wire b, $S_b^0 = S_a^1$ and $S_b^1 = S_b^0$.

From Boolean Circuit Queries to Set-Operation

Example: $f(x) = \bar{x}_1 \wedge \bar{x}_2 \vee (\neg \bar{x}_3)$ where $x = \bar{x}_1 \| \bar{x}_2 \| \bar{x}_3 \in \{0,1\}^3$



$$S_4^0 = S_1^0 \cup S_2^0$$
$$S_4^1 = S_1^1 \cap S_2^1$$
$$S_5^0 = S_3^1$$
$$S_5^1 = S_3^0$$
$$S_6^0 = S_4^0 \cap S_5^0 = S_1^0 \cup S_2^0 \cap S_3^1$$
$$S_6^1 = S_4^1 \cup S_5^1 = S_1^1 \cap S_2^1 \cup S_3^0$$

Output set:

$$S_{output} = S_6^1 = S_1^1 \cap S_2^1 \cup S_3^0$$

Zero-Knowledge Functional Elementary Databases

Zero-Knowledge Functional Elementary Databases

Setup(1^λ): Genrate using public parameters.

Commit(D): Let $S_i^b := \{x | (x, v) \in D \wedge \text{the i-th bit of "} x \| v \text{" is } b\}$

1. Use ZKS scheme to commit all S_i^b.
2. Use ZK-EDB scheme to commit D.

Prove(com, τ, f, D_{output}): Transform f into combined operation $\mathcal{Q}$,

1. Prove that for each $(x, v) \in D_{output}$ and each i, $x \in S_i^{\bar{x}_i}$ and $x \notin S_i^{1-\bar{x}_i}$.
   Showing the correctness of S_b^i.
2. Prove that $\{x | (x, v) \in D_{output}\} = \mathcal{Q}(S_1^0, S_1^1, \cdots)$.
   Showing the correctness of function.
3. Prove that for each $(x, v) \in D_{output}$, $(x, v) \in D$ through ZK-EDB.
   Showing the validness of associated value v.

Verify(com, f, D_{output}, π): Check the correctness of proofs.

Performance

Performance of our ZK-FEDB[1]:

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

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, $\mathbb{G}$ represents a group element, h denotes hashing to a prime and EXT is a λ-bit exponentiation.

[1]Utilizing our ZKS scheme and ZK-EDB scheme (constructed in the full version of our paper), and applying the standard batching technique.

Thank you for your attention