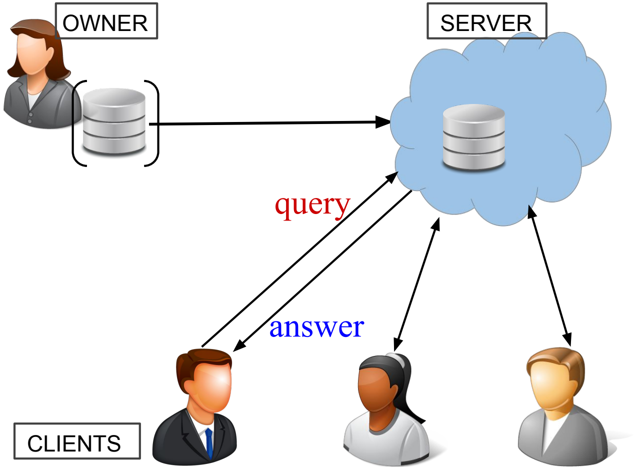
Zero Knowledge Accumulators and Set Operations

Esha Ghosh[1] Olya Ohrimenko[2] Dimitrios Papadopoulos[3]
Roberto Tamassia[1] Nikos Triandopoulos[4]

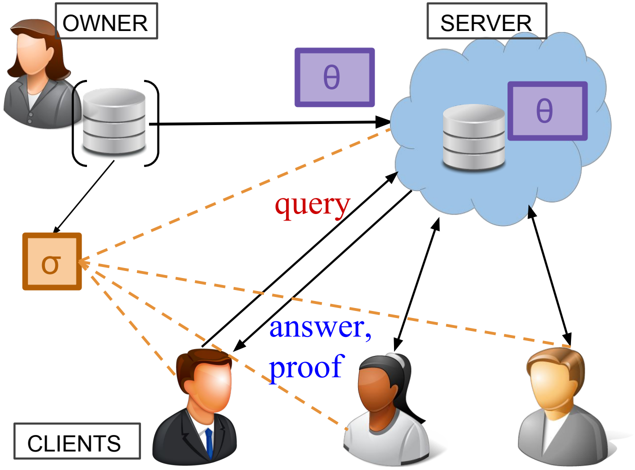
[1]Brown University [2]Microsoft Research
[3]University of Maryland [4]Stevens Institute of Technology

Research supported in part by the US National Science Foundation

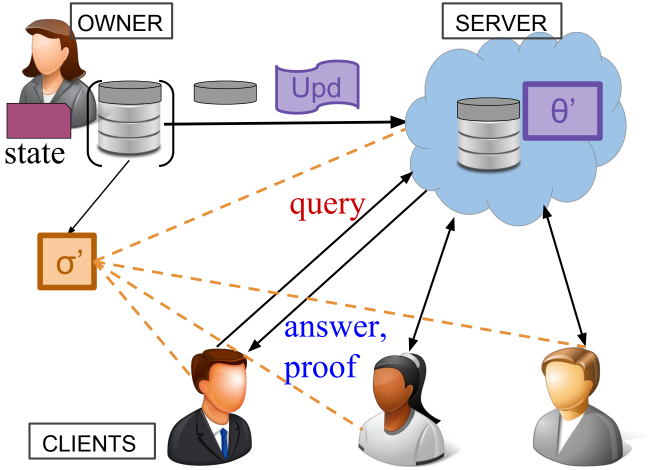
Data Outsourcing



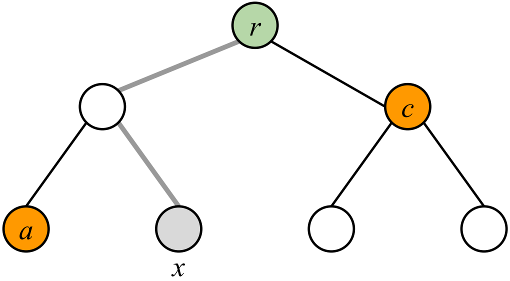
Verifiable data outsourcing (static)



Verifiable data outsourcing (dynamic)

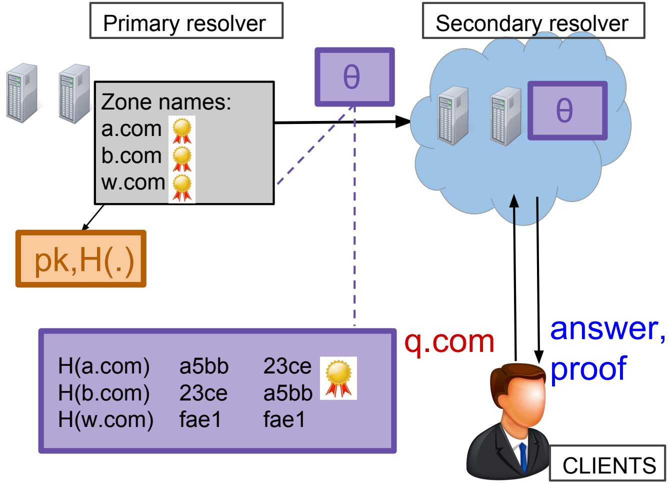


Challenge: Proof Leaking Information



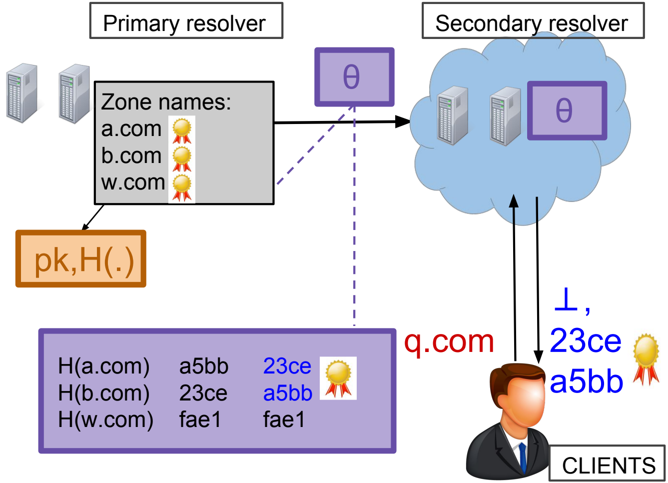
Merkle tree with items stored in sorted order at the leaves.
Proof of x: $((a, L), (b, R))$.
Verification: $h(h(a, h(x), c) = r$
Proof leaks rank of item.

Zone enumeration attack



[GNPRVZ– NDSS15, RFC 5155]

Zone enumeration attack



[Bernstein11–nsec3walker]

Cryptographic accumulator [Benaloh and del Mare93]

$\sigma \leftarrow \mathrm{acc}(\mathrm{Set}\mathcal{X})$.

Efficient and succinct proof for $x \in \mathcal{X}, x \notin \mathcal{X}$.

Proofs are publicly computable and verifiable.

Soundness: Forging proof for an element is infeasible.

Traditional proofs are leaky.

In this work

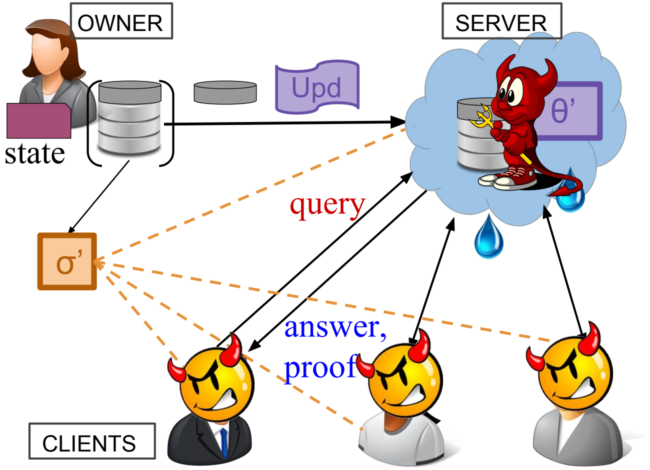
Formal model for zero-knowledge universal dynamic accumulators.

Efficient construction for zero-knowledge accumulators.

Efficient construction for :

1. is-subset
2. difference
3. union
4. intersection

Our Model



Soundness

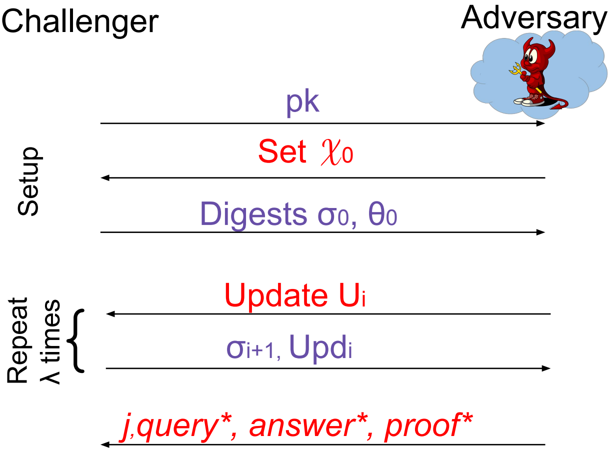


Figure: Probability that Verify accepts but *answer** is not correct wrt `query*` on $\mathcal{X}_j$ is negligible

Zero-Knowledge

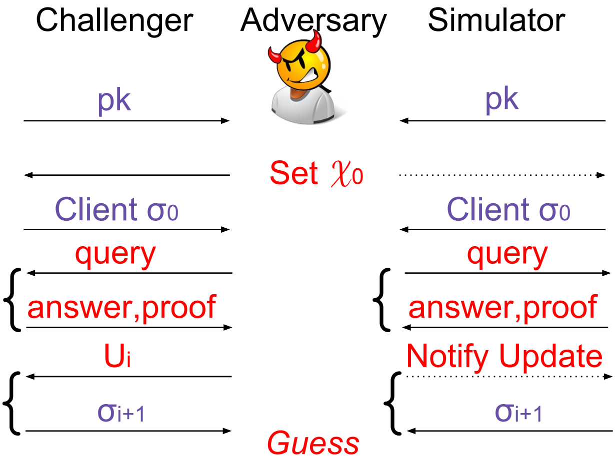


Figure: Probability that Adversary guesses correctly if it is talking to a challenger or a simulator is negligible

Zero Knowledge Accumulator

Query

$\mathcal{X} = \{x_1, \ldots, x_N\}$ = set of elements

Client Query: Is element $x \in \mathcal{X}$?

Server Response: answer = 1 indication yes and answer = 0 indicating no + proof

Set Representation

A set $\mathcal{X} = \{x_1, \ldots, x_N\}$ represented using its characteristic polynomial $\text{Ch}_{\mathcal{X}}[z] = \prod_{i=1}^{N}(z + x_i)$

Bilinear Map:

- $\lambda \in \mathbb{N}$ is the security parameter of the scheme
- G, G_1 multiplicative cyclic groups of prime order p
- p is a large k-bit prime
- g is a random generator of G
- $e : G \times G \rightarrow G_1$ is computable bilinear nondegenerate map
- $e(g^a, g^b) = e(g, g)^{ab}$.

Keygen and Setup (Owner)

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$

- Generate bilinear parameters $pub = (p, G, G_1, e, g)$.
  $O(\mathsf{poly}(\lambda))$
- Choose $s \xleftarrow{\$} \mathbb{Z}_p^*$.
- Set $\mathsf{sk} = s$ and $\mathsf{pk} = (g^s, pub)$.

$(\sigma_0, \theta_0, \mathsf{state}_0) \leftarrow \mathsf{Setup}(\mathsf{sk}, \mathcal{X}_0)$

- Choose $r \xleftarrow{\$} \mathbb{Z}_p^*$.
- Set $\sigma_0 = g^{r \cdot \mathsf{Ch}_{\mathcal{X}}(s)}$. $O(N)$
- Set $\theta_0 = (g, g^s, g^{s^2}, \ldots, g^{s^N}, r)$. $O(N)$
- Set $\mathsf{state}_0 = \mathcal{X}$.

Query (Server)

$(\text{answer}, \text{proof}) \leftarrow \text{PerformQuery}(\mathcal{X}_j, \theta_j, \texttt{query})$

- if $\texttt{query} = x \in \mathcal{X}$:

  set answer $= 1$ and proof $= (\sigma_j)^{\frac{1}{s+x}} = g^{\frac{r \cdot \text{Ch}_{\mathcal{X}}(s)}{(s+x)}}$. $O(N \log N)$

Query (Server)

$$\boxed{(\text{answer}, \text{proof}) \leftarrow \text{PerformQuery}(\mathcal{X}_j, \theta_j, \texttt{query})}$$

- if $\texttt{query} = x \in \mathcal{X}$:

  set answer $= 1$ and proof $= (\sigma_j)^{\frac{1}{s+x}} = g^{\frac{r \cdot \mathrm{Ch}_{\mathcal{X}}(s)}{(s+x)}}$. $\boxed{O(N \log N)}$
- if $\texttt{query} = x \notin \mathcal{X}$:

Query (Server)

$$(\text{answer}, \text{proof}) \leftarrow \text{PerformQuery}(\mathcal{X}_j, \theta_j, \texttt{query})$$

- if $\texttt{query} = x \in \mathcal{X}$:

  set answer $= 1$ and proof $= (\sigma_j)^{\frac{1}{s+x}} = g^{\frac{r \cdot \mathrm{Ch}_{\mathcal{X}}(s)}{(s+x)}}$. $O(N \log N)$

- if $\texttt{query} = x \notin \mathcal{X}$:

  1. Using the Extended Euclidean algorithm, compute polynomials $q_1[z], q_2[z]$ such that $q_1[z]\mathrm{Ch}_{\mathcal{X}}[z] + q_2[z](z + x) = 1$.
     $O(N \log^2 N \log \log N)$

Query (Server)

$$(\text{answer}, \text{proof}) \leftarrow \text{PerformQuery}(\mathcal{X}_j, \theta_j, \texttt{query})$$

- if $\texttt{query} = x \in \mathcal{X}$:
  set answer $= 1$ and proof $= (\sigma_j)^{\frac{1}{s+x}} = g^{\frac{r \cdot \mathsf{Ch}_{\mathcal{X}}(s)}{(s+x)}}$. $O(N \log N)$
- if $\texttt{query} = x \notin \mathcal{X}$:
  1. Using the Extended Euclidean algorithm, compute polynomials $q_1[z], q_2[z]$ such that $q_1[z]\mathsf{Ch}_{\mathcal{X}}[z] + q_2[z](z + x) = 1$.
     $O(N \log^2 N \log\log N)$
  2. Pick a random $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$

Query (Server)

$$(\text{answer}, \text{proof}) \leftarrow \text{PerformQuery}(\mathcal{X}_j, \theta_j, \texttt{query})$$

- if $\texttt{query} = x \in \mathcal{X}$:

  set answer $= 1$ and proof $= (\sigma_j)^{\frac{1}{s+x}} = g^{\frac{r \cdot \text{Ch}_{\mathcal{X}}(s)}{(s+x)}}$. $O(N \log N)$

- if $\texttt{query} = x \notin \mathcal{X}$:
  1. Using the Extended Euclidean algorithm, compute polynomials $q_1[z], q_2[z]$ such that $q_1[z]\text{Ch}_{\mathcal{X}}[z] + q_2[z](z + x) = 1$.
     $O(N \log^2 N \log \log N)$
  2. Pick a random $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$
  3. Set $q_1'[z] = q_1[z] + \gamma \cdot (z + x)$

Query (Server)

$$(\text{answer}, \text{proof}) \leftarrow \text{PerformQuery}(\mathcal{X}_j, \theta_j, \texttt{query})$$

- if $\texttt{query} = x \in \mathcal{X}$:

  set answer $= 1$ and proof $= (\sigma_j)^{\frac{1}{s+x}} = g^{\frac{r \cdot \mathsf{Ch}_{\mathcal{X}}(s)}{(s+x)}}$. $O(N \log N)$

- if $\texttt{query} = x \notin \mathcal{X}$:
  1. Using the Extended Euclidean algorithm, compute polynomials $q_1[z], q_2[z]$ such that $q_1[z]\mathsf{Ch}_{\mathcal{X}}[z] + q_2[z](z+x) = 1$.
     $O(N \log^2 N \log \log N)$
  2. Pick a random $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$
  3. Set $q_1'[z] = q_1[z] + \gamma \cdot (z+x)$
  4. Set $q_2'[z] = q_2[z] - \gamma \cdot \mathsf{Ch}_{\mathcal{X}}[z]$.

Query (Server)

$$\text{(answer, proof)} \leftarrow \text{PerformQuery}(\mathcal{X}_j, \theta_j, \texttt{query})$$

- if $\texttt{query} = x \in \mathcal{X}$:

  set answer $= 1$ and proof $= (\sigma_j)^{\frac{1}{s+x}} = g^{\frac{r \cdot \mathsf{Ch}_{\mathcal{X}}(s)}{(s+x)}}$. $O(N \log N)$

- if $\texttt{query} = x \notin \mathcal{X}$:
  1. Using the Extended Euclidean algorithm, compute polynomials $q_1[z], q_2[z]$ such that $q_1[z]\mathsf{Ch}_{\mathcal{X}}[z] + q_2[z](z+x) = 1$.

     $O(N \log^2 N \log \log N)$
  2. Pick a random $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$
  3. Set $q_1'[z] = q_1[z] + \gamma \cdot (z+x)$
  4. Set $q_2'[z] = q_2[z] - \gamma \cdot \mathsf{Ch}_{\mathcal{X}}[z]$.
  5. Set $W_1 := g^{q_1'(s)r^{-1}}, W_2 = g^{q_2'(s)}$.

Query (Server)

$(\text{answer}, \text{proof}) \leftarrow \text{PerformQuery}(\mathcal{X}_j, \theta_j, \texttt{query})$

- if $\texttt{query} = x \in \mathcal{X}$:

  set answer $= 1$ and proof $= (\sigma_j)^{\frac{1}{s+x}} = g^{\frac{r \cdot \text{Ch}_{\mathcal{X}}(s)}{(s+x)}}$. $O(N \log N)$

- if $\texttt{query} = x \notin \mathcal{X}$:

  1. Using the Extended Euclidean algorithm, compute polynomials $q_1[z], q_2[z]$ such that $q_1[z]\text{Ch}_{\mathcal{X}}[z] + q_2[z](z+x) = 1$.

     $O(N \log^2 N \log \log N)$

  2. Pick a random $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$
  3. Set $q_1'[z] = q_1[z] + \gamma \cdot (z+x)$
  4. Set $q_2'[z] = q_2[z] - \gamma \cdot \text{Ch}_{\mathcal{X}}[z]$.
  5. Set $W_1 := g^{q_1'(s) r^{-1}}, W_2 = g^{q_2'(s)}$.
  6. Set proof $:= (W_1, W_2)$ and answer $= 0$.

Verification (Client)

$$\boxed{(\text{accept}/\text{reject}) \leftarrow \text{Verify}(\text{pk}, \sigma_j, \texttt{query}, \text{answer}, \text{proof})}$$

- Let $\texttt{query} = x$.
- If answer $= 1$, return accept if $e(\sigma_j, g) = e(\text{proof}, g^x \cdot \text{pk})$. $\boxed{O(1)}$
- if answer $= 0$, return accept if $e(W_1, \sigma_j)e(W_2, g^x \cdot \text{pk}) = e(g, g)$. $\boxed{O(1)}$
- Return reject otherwise.

Update

$$(\mathcal{X}_{i+1}, \sigma_{i+1}, \mathsf{upd}_i, \mathsf{state}_{i+1}) \leftarrow \mathsf{Update}(\mathsf{sk}, \mathsf{state}_i, \sigma_i, \theta_i, \mathcal{X}_i, u_i)$$

Owner:

- Choose $r' \xleftarrow{\$} \mathbb{Z}_p^*$.
- If x is to be inserted:
  1. Compute $\sigma_{i+1} = \sigma_i^{(s+x)r'}$. $O(1)$
- If x is to be deleted:
  1. Compute $\sigma_{i+1} = \sigma_i^{\frac{r'}{s+x}}$. $O(1)$
- Set $\mathsf{upd}_i = (r')$ and $\mathsf{state}_{i+1} = \mathcal{X}_{i+1}$.

Server:

Store the inserted/deleted element and $\mathsf{upd}_i = (r')$. $O(1)$

Privacy comes almost for free

	[Nguyen05 – No Privacy]	This work
Setup	NMUL	NMUL
Update	1MUL	2MUL
Witness (Member)	NMUL $+ (N-1)$ADD	NMUL $+ (N-1)$ADD
Witness (Non-Member)	NMUL $+ (N-1)$ADD	$(N+1)$MUL $+ (N-1)$ADD
Verify (Member)	1(MUL + ADD + PAIR)	1(MUL + ADD + PAIR)
Verify (Non-Member)	2(MUL + ADD + PAIR)	1(MUL + ADD + ADD_1) + 2PAIR
Witness Update (Member)	1(MUL + ADD)	2MUL + 1ADD
Witness Update (Non-Member)	2MUL + 1ADD	$(N+1)$MUL $+ (N-1)$ADD

Figure: ADD = point addition MUL = scalar multiplication in the elliptic curve group G, ADD_1 = point addition in G_1 and PAIR a pairing computation, whereas N is the size of the set.

Set Algebra : Union

Query

$\{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ = set collection

Client Query: Return union of sets $2, 5, 9$

Server Response: answer $= \mathcal{X}_2 \cup \mathcal{X}_5 \cup \mathcal{X}_9 +$ proof

Let $\mathcal{X}_2 = \{a, b, d\}, \mathcal{X}_5 = \{d, f\}, \mathcal{X}_9 = \{a, c\}$

answer $= \{a, c, b, d, f\}$

Completeness Conditions

Superset condition: $\mathcal{X}_2 \subseteq \text{answer} \wedge \mathcal{X}_5 \subseteq \text{answer} \wedge \mathcal{X}_9 \subseteq \text{answer}$.

Technique: Generalization of set membership.

Membership condition: $\text{answer} \subseteq \tilde{U}$ where $\tilde{U} = \mathcal{X}_2 \uplus \mathcal{X}_5 \uplus \mathcal{X}_9$.

Proving membership

Multiset union: $\tilde{U} = \{a, a, c, c, b, d, d, f\}$

1. Compute
   $\sigma_{\tilde{U}} \leftarrow g^{(r_2 r_5 r_9)\mathsf{Ch}_{\tilde{U}}(s)} = g^{(r_2 r_5 r_9)(s+a)^2(s+c)(s+b)(s+d)^2(s+f)}$

Proving membership

Multiset union: $\tilde{U} = \{a, a, c, c, b, d, d, f\}$

1. Compute
   $\sigma_{\tilde{U}} \leftarrow g^{(r_2 r_5 r_9)\mathrm{Ch}_{\tilde{U}}(s)} = g^{(r_2 r_5 r_9)(s+a)^2(s+c)(s+b)(s+d)^2(s+f)}$
2. Prove $\sigma_{\tilde{U}}$ is correctly computed

Proving membership

Multiset union: $\tilde{U} = \{a, a, c, c, b, d, d, f\}$

1. Compute
   $\sigma_{\tilde{U}} \leftarrow g^{(r_2 r_5 r_9)\mathrm{Ch}_{\tilde{U}}(s)} = g^{(r_2 r_5 r_9)(s+a)^2(s+c)(s+b)(s+d)^2(s+f)}$
2. Prove $\sigma_{\tilde{U}}$ is correctly computed
3. Prove answer $\subseteq \tilde{U}$ using $\sigma_{\tilde{U}}$

Step 2: Server

$\sigma_9 = g^{r_9(s+a)(s+c)}$

$\sigma_2 = g^{r_2(s+a)(s+b)(s+d)}$

$\sigma_5 = g^{r_5(s+d)(s+f)}$

Step 2: Server

$$\sigma_{\tilde{U}} = g^{(r_2 r_5 r_9)(s+a)^2(s+c)(s+b)(s+d)^2(s+f)}$$

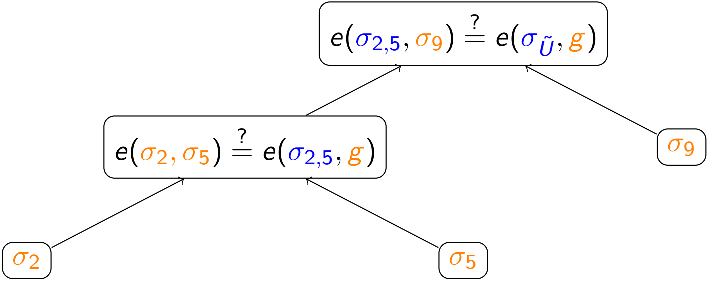
$$\sigma_{2,5} = g^{r_2 r_5 (s+a)(s+b)(s+d)^2(s+f)}$$

$$\sigma_9 = g^{r_9 (s+a)(s+c)}$$

$$\sigma_2 = g^{r_2 (s+a)(s+b)(s+d)}$$

$$\sigma_5 = g^{r_5 (s+d)(s+f)}$$

Step 2: Client



Step 3

Server: $W_{(\text{answer},\tilde{U})} \leftarrow g^{\frac{r_2 r_5 r_9 \mathrm{Ch}_{\tilde{U}}(s)}{\mathrm{Ch}_{\text{answer}}(s)}} = g^{r_2 r_5 r_9 (s+a)(s+d)}$

Client: $e(W_{(\text{answer},\tilde{U})}, g^{\mathrm{Ch}_{\text{answer}}(s)}) \stackrel{?}{=} e(\sigma_{\tilde{U}}, g)$

More in the paper:

1. Relation of Zero Knowledge Accumulator with the existing primitives (ZKS, PSR, Trapdoorless Acc).
2. Formal proof that Zero knowledge is stronger than indistinguishably notion [MLPP12, DHS15] of privacy.
3. First efficient construction for zero-knowledge verifiable set algebra queries (Is-subset, Intersection, Union, Difference) with no additional cost over the state-of-the art non-private construction [PTT11].

Thank you!