Zero Knowledge Sets with short proofs

Dario Catalano Dario Fiore Mariagrazia Messina[1]

Dipartimento di Matematica ed Informatica – Università di Catania, Italy

April 16, 2008

EUROCRYPT 2008 - Istanbul

[1] Now in Microsoft Italia

Outline

Zero Knowledge sets

Parties

- A prover $\mathcal{P}$
- A verifier $\mathcal{V}$

The problem

- $\mathcal{P}$ knows a finite secret set S
- $\mathcal{V}$ is allowed to ask $\mathcal{P}$ questions of the form: "$x \in S$" or "$x \notin S$"
- $\mathcal{P}$ answers such questions by providing publicly verifiable proofs

Informal requirements

- The proofs should not reveal any further information (i.e. not even the size of S)
- The proofs should be reliable
  - A cheating $\mathcal{P}$ cannot convince $\mathcal{V}$ that some element x is in the set while is not (or viceversa).
  - $\mathcal{V}$ learns about S only membership or non membership of elements.

Zero Knowledge EDB - Formal definition

- The problem was first defined by [MRK03].
- More precisely they defined *Zero Knowledge Elementary Databases* (EDBs)
- Notation
  - Let D be a database, x a DB key
  - $D(x) = y$: if y is the database value associated to x
  - $D(x) = \perp$: if $x \notin D$.

Elementary Databases

Formally, an EDB system is defined by a triple of algorithms:

- $Commit(CRS, D) \rightarrow (ZPK, ZSK)$ $//D$ database, CRS common reference string
- $Prove(CRS, ZSK, x) \rightarrow (\pi_x)$ // x DB key, π_x proof of either $D(x) = y$ or $D(x) = \perp$
- $Verify(CRS, ZPK, x, \pi_x)$ outputs y if $D(x) = y$, out if $D(x) = \perp$ or $\perp$ if π_x is not valid.

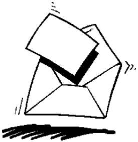
Zero Knowledge EDBs - Requirements

1. *Completeness*. Proofs created by a honest prover are correct.
2. *Soundness.* A dishonest prover cannot produce two different proofs for the same value, that are both valid.
3. *Zero-Knowledge.* Proofs do not reveal any information except membership or not membership.

"ZKS story"

- [MRK03] proposed a construction of ZKS by using a variant of the Pedersen's Commitment in the CRS
- Later [CHMLR05] showed that:
  - such variant is an instantiation of a new type of commitments: "*mercurial commitments*"
  - mercurial commitments can be used as building block for ZKS
  - mercurial commitments can be built from general assumptions (i.e. NIZK)
- Finally [CDV06] gave a construction of mercurial commitments from one way functions in the CRS
- This result showed that ZKS are equivalent to collision resistant hash functions in the CRS

Commitment scheme



- Digital equivalent of an opaque envelop.

1. *Hiding property.* Whatever is put inside the envelop remain secret until the latter is opened.
2. *Binding property.* Whoever creates the commitment should not be able to open it with a message that is not the one originally inserted

- Example: Perdersen's commitment (based on discrete log).

Mercurial commitments

- [CHMLR05] introduced mercurial commitments and defined their properties
- A mercurial commitment can be created *hard* or *soft*.
- Two decommiting produres: *hard-opening*, *soft-opening*.
- Hard commitments are like standard ones:
  - they can be hard/soft-opened only with respect to the message used to construct the commitment
- Soft commitments can be soft-opened to any message, but they cannot be hard opened.

Mercurial commitments - Properties

- They satisfy slightly different binding and hiding properties according to the new definition:
  - *Mercurial binding*
  - *Mercurial hiding*: it is infeasible to distinguish hard commitments from soft ones

MRK scheme

Construction by [MRK03] with the generalization by Chase *et al.* using mercurial commitments.

- Use an authenticated Merkle tree of depth k.
- Each leaf is related to a DB key x and contains the commitment to $D(x)$ (or to 0 if $D(x) = \perp$)
- Each node is a mercurial commitment of its two children.
- The root ϵ contains the commitment of the tree (ZKS PK).

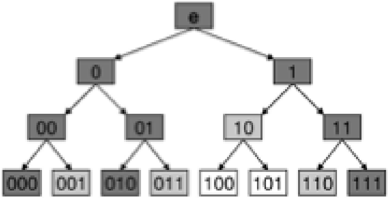


Figure: The complete labeled binary tree of depth 3 for $S = \{000, 010, 111\}$. The light shaded vertices comprise $FRONTIER(S)$.

MRK scheme (2)

- To prove that $x \in \{0,1\}^k$ belongs to the committed set S, the prover opens all the commitments in the path from the root ϵ to the leaf labeled by x.
- Verification: verify each commitment in the path.

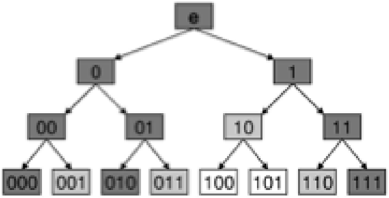


Figure: The complete labeled binary tree of depth 3 for $S = \{000, 010, 111\}$. The light shaded vertices comprise *FRONTIER*(S).

MRK scheme (3)

- **It is not necessary to generate the complete binary tree.**
- Prune the tree by cutting those subtrees containing only keys of elements not in the database.
- The roots of such subtrees are kept in the tree ("frontier").
- Frontier nodes contain soft commitments "to nothing".

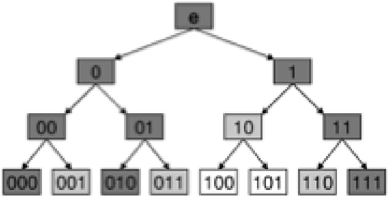


Figure: The complete labeled binary tree of depth 3 for $S = \{000, 010, 111\}$. The light shaded vertices comprise $FRONTIER(S)$.

MRK scheme (4)

- Upon receiving a query for $x \notin S$, the missing subtree containing x is generated on-line.
- **Soft commitments in the frontier nodes are then soft-opened to the values contained in its newly generated children.**

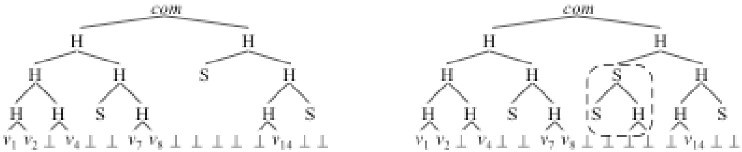


Figure: A commitment tree before and after a query for key 101, whose value is not the DB. The parts built in response to the query are shown in the second tree. Hard commitments are denoted by H and soft commitments by S.

Motivating question

Assumptions to construct ZKS are well studied

What about practical solutions?

In the MRK scheme verification time and proof length are linear in $log_2(2^k)$ (for $x \in \{0,1\}^k$).

Motivating question

Assumptions to construct ZKS are well studied

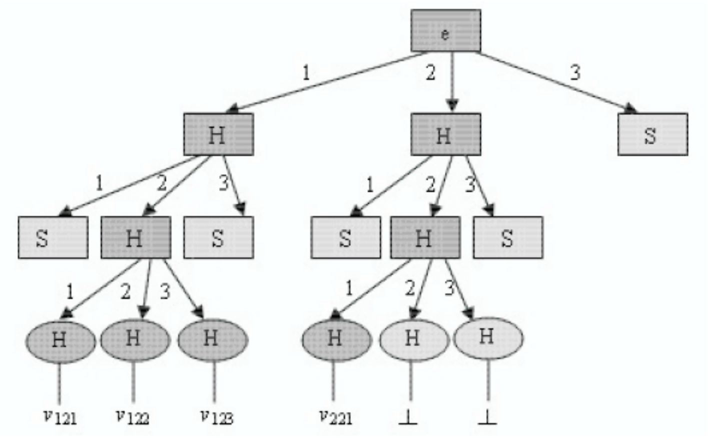
What about practical solutions?

In the MRK scheme verification time and proof length are linear in $log_2(2^k)$ (for $x \in \{0,1\}^k$).

Idea:

Reducing tree height by increasing the branching factor of the tree

Result: a q-ary tree



The trivial solution

MRK with q-ary trees

Issues:

- For a correct authentication we need to give all the siblings for each level
- Proof length remains the same as in MRK

Solution: *q*-mercurial commitments

- We propose a new primitive called "*trapdoor q-mercurial commitment*" (qTMC)
- We prove that ZKS can be constructed from qTMC
- qTMC allows to commit to an (ordered) sequence of q messages
- The binding property keeps in consideration the position of each message in the sequence.

qTMC construction from SDH assumption

We propose a construction based on the Strong Diffie-Hellman assumption (SDH) [BB04].

SDH assumption

Informally, the SDH assumption in bilinear groups G_1, G_2 of prime order p states that, for every PPT algorithm $\mathcal{A}$ and for a parameter q, the following probability is negligible:

$$Pr[\mathcal{A}(g_1, g_1^x, g_1^{(x^2)}, \cdots, g_1^{(x^q)}, g_2, g_2^x) = (c, g_1^{1/(x+c)})].$$

qTMC construction (sketch)

- The construction is inspired to the simulator of the Boneh-Boyen weak signature scheme.
- $PK = (A_0 = g_1, A_1 = g_1^x, \cdots, A_q = g_1^{x^q}, g_2, h = g_2^x), TK = x$
- $\mathrm{qHCom}(m_1, \cdots, m_q)$.
  - $C_i = H(i||m_i)$ binds each message with its position.
  - Define $f(z) = \prod_{i=1}^{q}(z + C_i)$. Extract β_i coefficients. Pick α random. Let $\gamma = \alpha x$
  - Set $g_1' = g_1^{f(\alpha x)} = \prod_{i=0}^{q} A_i^{\beta_i \alpha^i}$, $g_2' = g_2^{\gamma} = h^{\alpha}$.
  - The commitment is $C = (g_1', g_2')$ (*similar to BB simulator's* PK)

qTMC construction (sketch)

- $\mathsf{qHOpen}_{PK}(m, j, \mathtt{aux})$. Output all values needed to reconstruct the commitment. $(\alpha, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$.
- $\mathsf{qSCom}_{PK}()$. Create random values g_1', g_2'. Pick random $\alpha', y \leftarrow \mathbb{Z}_p^*$, set $g_1' = g_1^{\alpha'}, g_2' = g_2^y$. Output $C = (g_1', g_2')$.
- $\mathsf{qSOpen}_{PK}(m, j, \mathtt{flag}, \mathtt{aux})$
  - If $\mathtt{flag} = \mathbb{H}$. Define $f_j(z) = \frac{f(z)}{(z+C_j)} = \prod_{i=1 \wedge i \neq j}^{q} (z + C_i) = \sum_{i=0}^{q-1} \delta_i z^i$. Compute $\sigma_j = (g_1')^{\frac{1}{\gamma + C_j}} = g_1^{\frac{f(\gamma)}{\gamma + C_j}} = \prod_{i=0}^{q-1} A_i^{\delta_i \alpha^i}$. (*similar to BB simulator's signature extraction*)
  - If $\mathtt{flag} = \mathbb{S}$ output $\sigma_j = (g_1')^{\frac{1}{y + C_j}}$.

qTMC construction

- $\mathsf{qSVer}_{PK}(m, j, C, \tau)$ // $C = (g_1', g_2'), \tau = \sigma_j$
  Check if $e(\sigma_j, g_2' g_2^{C_j}) = e(g_1', g_2)$.

Correctness

If $\sigma_j = (g_1')^{\frac{1}{\gamma + C_j}}$ then $e((g_1')^{\frac{1}{\gamma + C_j}}, g_2^{\gamma} g_2^{C_j}) = e(g_1', g_2)$

qTMC construction

- $\mathrm{qSVer}_{PK}(m, j, C, \tau)$ // $C = (g'_1, g'_2), \tau = \sigma_j$
  Check if $e(\sigma_j, g'_2 g_2^{C_j}) = e(g'_1, g_2)$.

Correctness

If $\sigma_j = (g'_1)^{\frac{1}{\gamma + C_j}}$ then $e((g'_1)^{\frac{1}{\gamma + C_j}}, g_2^{\gamma} g_2^{C_j}) = e(g'_1, g_2)$

Efficiency of qTMC

- Size of each hard opening still depends linearly on q.
- Size of each soft opening is *indipendent* of q // $\Theta(1)$!

ZKS from qTMC - Results

Table: Length of the proofs (expressed as number of group elements) in the case of $k = 128$ bits of security

	Membership	**Non-membership**
MRK scheme	773	644
Our scheme $(q = 8)$	517 (*33% shorter*)	175 (*73% shorter*)

Conclusions and open problems

- Our work introduces a new primitive called *q*-mercurial commitment (qTMC)
- qTMCs are used to improve the construction of zero-knowledge sets in terms of proofs length
- Interesting challenges:
  - to construct more efficient qTMCs
  - in particular to construct a qTMC that allows for hard-openinings with lenght independent of *q*

Thanks!