# Zero-Knowledge Sets with short proofs*

Dario Catalano[1], Dario Fiore[1], and Mariagrazia Messina[2]**

[1] Dipartimento di Matematica ed Informatica – Università di Catania, Italy.
{catalano,fiore}@dmi.unict.it
[2] Microsoft Italia. mariame@microsoft.com

**Abstract.** Zero Knowledge Sets, introduced by Micali, Rabin and Kilian in [17], allow a prover to commit to a secret set $S$ in a way such that it can later prove, non interactively, statements of the form $x \in S$ (or $x \notin S$), without revealing any further information (on top of what explicitly revealed by the inclusion/exclusion statements above) on $S$, not even its size. Later, Chase *et al.* [5] abstracted away the Micali, Rabin and Kilian's construction by introducing an elegant new variant of commitments that they called (trapdoor) mercurial commitments. Using this primitive, it was shown in [5, 4] how to construct zero knowledge sets from a variety of assumptions (both general and number theoretic).

In this paper we introduce the notion of trapdoor $q$-mercurial commitments ($\mathsf{qTMC}$s), a notion of mercurial commitment that allows the sender to commit to an *ordered* sequence of exactly $q$ messages, rather than to a single one. Following [17, 5] we show how to construct ZKS from $\mathsf{qTMC}$s and collision resistant hash functions.

Then, we present an efficient realization of $\mathsf{qTMC}$s that is secure under the so called Strong Diffie Hellman assumption, a number theoretic conjecture recently introduced by Boneh and Boyen in [3]. Using our scheme as basic building block, we obtain a construction of ZKS that allows for proofs that are much shorter with respect to the best previously known implementations. In particular, for an appropriate choice of the parameters, our proofs are up to 33% shorter for the case of proofs of membership, and up to 73% shorter for the case of proofs of non membership.

## 1 Introduction

Imagine some party $P$ wants to commit to a set $S$, in a way such that any other party $V$ can "access" $S$ in a limited but reliable manner. By limited here we mean that $V$ is given indirect access to $S$, in the sense that she is allowed to ask only questions of the form "is $x$ in $S$?". $P$ answers such questions by providing publicly verifiable proofs for the statements $x \in S$ (or $x \notin S$). Such proofs should be reliable in the sense that a cheating $P$ should not be able to convince $V$ that some $x$ is in the set while is not (or viceversa). At the same time, they should be "discreet" enough not to reveal anything beyond their validity.

* The full version of this paper is available at http://www.dmi.unict.it/~fiore

** Work entirely done while student at University of Catania

The notion of Zero Knowledge Sets (ZKS) was recently introduced by Micali, Rabin and Kilian [17] to address exactly this problem. Informally, ZKS allow a prover $P$ to commit to an arbitrary (but finite) set $S$ in a way such that $P$ can later prove statements of the form $x \in S$ or $x \notin S$ without revealing any significant information about $S$ (not even its size!). As already pointed out in [17], the notion of zero knowledge sets can be easily extended to encompass the more general notion of elementary databases (EDB). In a nutshell, an elementary database is a set $S$ with the additional property that each $x \in S$ comes with an associated value $S(x)$. In the following we will refer to ZKS to include zero knowledge EDB as well.

The solution by Micali *et al.* is non interactive and works in the so called *shared random string* model (i.e. where a random string, built by some trusted third party, is made available to all participants) building upon a very clever utilization of a simple commitment scheme, originally proposed by Pedersen [20].

Commitment schemes play a central role in cryptography. Informally, they can be seen as the digital equivalent of an opaque envelope. Whatever is put inside the envelope remains secret until the latter is opened (hiding property) and whoever creates the commitment should not be able to open it with a message that is not the one originally inserted (binding property). Typically, a commitment scheme is a two phase procedure. During the first phase, the sender creates a commitment $C$, to some message $m$, using an appropriate commitment algorithm, and sends $C$ to the receiver $R$. In the opening phase the sender opens $C$ by giving $R$ all the necessary material to (efficiently) verify that $C$ was indeed a valid commitment to $m$.

Since Pedersen's commitment relies on the intractability of the discrete logarithm, so does the construction in [17]. Later, Chase *et al.* [5] abstracted away Micali *et al.*'s solution and described the exact properties a commitment scheme should possess in order to allow a similar construction. This led to an elegant new variant of commitments, that they called *mercurial commitment*.

Informally, a mercurial commitment is a commitment scheme where the binding requirement is somewhat relaxed in order to allow for two decommitment procedures: an *hard* and a *soft* one. At committing time, the sender can decide as whether to create an *hard* commitment or a *soft* one, from the message $m$ he has in mind. Hard commitments are like standard ones, in the sense that they can be (hard or soft) opened only with respect to the message that was originally used to construct the commitment. Soft commitments, on the other hand, allow for more freedom, as they cannot be hard opened in any way, but they can be soft opened to any arbitrary message. An important requirement of mercurial commitments is that, hard and soft commitments should look alike to any polynomially bounded observer.

Using this new primitive, Chase *et al.* proved that it is possible to construct ZKS from a variety of assumptions (number theoretic or general) [3]. Their most general implementation, shows that (non interactive) ZKS can be constructed,

[3] More precisely, they require the mercurial commitment to be *trapdoor* as well. Very informally, this means that the scheme comes with a trapdoor information *tk* (nor-

in the shared random string model, assuming non interactive zero knowledge proofs (NIZK) [2] and collision resistant hash functions [8] [4]. Moreover, they showed that collision resistant hash function are necessary to construct ZKS, as they are implied by the latter. Finally, Catalano, Dodis and Visconti [4] gave a construction of (trapdoor) mercurial commitments from one way functions in the shared random string model. This result completed the picture as it showed that collision resistant hash functions are necessary and sufficient for non interactive ZKS in the shared random string model.

Our Contribution. All the constructions above, build upon the common idea of constructing an authenticated Merkle tree of depth $k$ where each internal node is a mercurial commitment (rather than the hash) of its two children. Very informally, to prove that a given $x \in \{0,1\}^k$ belongs to the committed set $S$, the prover simply opens all the commitments in the path from the root to the leaf labeled by $x$ (more details about this methodology will be given later on). Thus the length of the resulting proof is $k \cdot d$, where $d$ denotes the length of the opening of the mercurial commitment, and $k$ has to be chosen so that $2^k$ is larger than the size of any "reasonably" large set $S$ [5]. Assuming $k = 128$ and $d = O(k)$, as it is the case for all known implementations, this often leads to very long proofs.

It is thus important to research if using the properties of specific number-theoretic problems, it is possible to devise zero knowledge sets that allow for shorter proofs. Such proofs would be desirable in all those scenarios where space or bandwidth are limited. A typical example of such a scenario is mobile internet connections, where customers pay depending on the number of blocks sent and received.

In this paper, we present a new construction of ZKS that allows for much shorter proofs, with respect to the best currently known implementation (which is the Micali *et al.* construction when implemented on certain classes of elliptic curves. From now on we will use the acronym MRK to refer to such an implementation).

Our solution relies on a new primitive that we call *trapdoor q-Mercurial Commitment* (qTMC, for short). Informally, qTMCs allow the sender to commit to a *sequence* of exactly $q$ messages $(m_1, \ldots, m_q)$, rather than to a single one, as with standard mercurial commitments. The sender can later open the commitment with respect to any message $m_i$ but, in order to do so successfully, he has to correctly specify the exact position held by the message in the sequence. In other

mally not available to anyone) that allows to completely destroy the binding property of the commitment

[4] It is known that one can construct NIZK under the assumption that trapdoor permutations exist or under the assumption that verifiable random functions (VRF) exist [12, 18]. These two assumptions, however, are, as far as we currently know, incomparable.

[5] This is because $2^k$ is also an upper bound for the size of the set. Thus, to meet the requirements of ZKS it should not reveal anything about the cardinality of the set itself.

words, trapdoor q-Mercurial commitments allow to commit to *ordered* sequences of $q$ messages.

Following [17, 5], we show how to construct ZKS from $\mathsf{qTMC}$s and collision resistant hash functions. This step is rather simple but very useful for our goal, as it reduces the task of realizing efficient ZKS to the task of realizing efficient $\mathsf{qTMC}$s. Indeed, even though the proposed transformation allows us to use a "flat" Merkle tree (i.e. with branching factor $q$, rather than two), it *does not* lead, by itself, to shorter proofs.

Recall that, informally, a proof for the statement $x \in S$ (or $x \notin S$) consists of an authenticated path from the root to the leaf labeled by $x$. The trouble is that in all known implementations of ZKS, to verify the authenticity of a node in the path, one must know all siblings of the node. If the tree is binary, the proof contains twice as many nodes as the the depth of the tree (since each node must be accompanied by its sibling). Thus, the length of a proof being proportional to the branching factor of the tree, increasing the latter, is actually a *bad* idea in general. Indeed, suppose we want to consider sets defined over a universe of $N$ elements. Using a binary authentication tree one gets proofs whose length is proportional to $\log_2 N(2n)$, where $n$ is the size of the authentication information contained in each node. Using a tree with branching degree $q$, on the other hand, one would get proofs of size $\log_q N(qn)$, which is actually more than in previous case.

Overcoming the proofs blow-up. In this paper we propose an implementation of trapdoor $q$ mercurial commitments that overcomes the above limitation. Our solution relies on the so called Strong Diffie Hellman assumption originally introduced by Boneh and Boyen [3] and builds upon the weakly secure digital signature given in [3]. The proposed implementation exploits the algebraic properties of the employed number theoretic primitive to produce a $\mathsf{qTMC}$ that allows for short openings. More precisely the size of each hard opening still depends linearly on $q$, but the size of each soft opening becomes constant and completely *independent* of $q$.

This results in ZKS that allow for much shorter proofs than MRK. Concretely, and for an appropriate choice of the parameter $q$, our proofs are up to 33% shorter for the case of proofs of membership, and up to 73% shorter for the case of proofs of non membership.

Zero Knowledge Sets vs Signatures. The idea of obtaining short proofs by changing the authentication procedure to deal with a "flat" authentication tree, is reminiscent of a technique originally suggested by Dwork and Naor [9], in the context of digital signature schemes. In a nutshell, the Dwork-Naor method allows to increase the branching factor of the tree without inflating the signature size. This is achieved, by, basically, authenticating each node with respect to its parent, but without providing its siblings.

Adapting this idea to work to the case of zero knowledge sets, presents several non trivial technical difficulties[6]. The main problem comes from the fact that, in ZKS, one has to make sure that a dishonest prover cannot construct two, both valid, proofs for the statements $x \in S$ and $x \notin S$. Such a requirement imposes limitations just not present when dealing with digital signatures [7].

Other Related work. Ostrovsky, Rackoff and Smith [19] described a construction that allows a prover to commit to a database and to provide answers that are consistent with the commitment. Their solution can handle more elaborate queries than just membership ones. Moreover they also consider the issue of adding privacy to their protocol. However their construction requires interaction (at least if one wants to avoid the use of random oracles) and requires the prover to keep a counter for the questions asked so far.
Gennaro and Micali [11] recently introduced the notion of independent zero knowledge sets. Informally, independent ZKS protocols prevent an adversary from successfully correlate her set to the one of a honest prover. Their notion of independence also implies that the resulting ZKS protocol is non-malleable and requires a new commitment scheme that is both independent and mercurial. We do not consider such an extension here.
Liskov [15] considered the problem of updating zero-knowledge databases. In [15] definitions for updatable zero knowledge databases are given, together with a construction based on verifiable random functions [18] and mercurial commitments. The construction, however, is in the random oracle model [1].
Very recently Prabhakaran and Xue [21] introduced the notion of statistically hiding sets (SHS) that is related but different than ZKS. Informally, SHS require the hiding property to hold with respect to unbounded verifiers. At the same time, however, they relax the zero knowledge requirement to allow for unbounded simulators.

Road Map. The paper is organized as follows. In section 2 we introduce the notion of trapdoor $q$ mercurial commitments and provide the relevant definitions for zero knowledge sets. Section 3 is devoted to the construction of ZKS from trapdoor $q$ mercurial commitments. In section 4 we show how to construct efficient qTMCs from the Strong Diffie Hellman Assumption. Efficiency considerations and comparisons with previous work are given in section 5. Finally conclusions and directions for future work are given in section 6

[6] It is probably instructive to mention the fact that, indeed, the Dwork Naor solution, and its improvements such as [7], *do not* work in our setting

[7] For instance, the soundness requirement above, imposes that exactly one single path from the root to a leaf, should be "labelable" as $x$. It seems very hard (if at all possible) to achieve this, when both type of proofs (i.e. proofs of membership and proofs of non membership) allow to authenticate each node (with respect to its parent), without providing its siblings.

## 2 Preliminaries

Informally, we say that a function is *negligible* if it vanishes faster than the inverse of any polynomial.

### 2.1 Trapdoor q-mercurial commitments

Informally, a trapdoor q-mercurial commitment ($\mathsf{qTMC}$ for brevity) extends the notion of (trapdoor) mercurial commitment, by allowing the sender to commit to an (ordered) sequence of $q$ messages, rather than to a single one. More precisely, and like standard (trapdoor) mercurial commitments (see [4] for a definition of trapdoor mercurial commitments), trapdoor q-mercurial commitments allows for two different decommitting procedures. In addition to the standard opening mechanism, there is a partial opening (also referred as *tease* or soft open) algorithm that allows for some sort of equivocation. At committing stage, the sender can decide to produce a commitment in two ways. Hard commitments should be hiding in the usual sense, but should satisfy a very strong binding requirement (that we call strong binding). Informally, strong binding means that a sender $S$ should be able to open a commitment only with respect to messages that were in the "correct" position in the sequence $S$ originally committed to. More precisely, when opening an hard commitment for a message $m$, the sender is required to specify an index $i \in \{1, \ldots, q\}$, indicating the position of $m$ in the sequence. In the case of hard commitments, the strong binding property imposes that the commitment should be successfully opened and teased to $(m, i)$ only if $m$ was the $i$-th message in the sequence $S$ originally committed to. Soft commitments, on the other hand cannot be opened, but can be teased with respect to messages belonging to any arbitrary sequence of $q$ messages.

More formally, a trapdoor $q$-mercurial commitment is defined by the following set of algorithms: ($\mathsf{qKeyGen}, \mathsf{qHCom}, \mathsf{qHOpen}, \mathsf{qHVer}, \mathsf{qSCom}, \mathsf{qSOpen}, \mathsf{qSVer}, \mathsf{qFake}, \mathsf{qHEquiv}, \mathsf{qSEquiv}$).

$\mathsf{qKeyGen}(1^k, q)$ is a probabilistic algorithm that takes in input a security parameter $k$ and the number $q$ of committed values and outputs a pair of public/private keys $(pk, tk)$.

$\mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$ Given an ordered tuple of messages, $\mathsf{qHCom}$ computes a hard commitment $C$ to $(m_1, \cdots, m_q)$ using the public key $pk$ and returns some auxiliary information $\mathsf{aux}$.

$\mathsf{qHOpen}_{pk}(m, i, \mathsf{aux})$ Let $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$, if $m = m_i$ the hard opening algorithm $\mathsf{qHOpen}_{pk}(m, i, \mathsf{aux})$ produces a hard decommitment $\pi$. The algorithm returns an error message otherwise.

$\mathsf{qHVer}_{pk}(m, i, C, \pi)$ The hard verification algorithm $\mathsf{qHVer}_{pk}(m, i, C, \pi)$ accepts (outputs 1) only if $\pi$ proves that $C$ is created to a tuple $(m_1, \cdots, m_q)$ such that $m_i = m$.

$\mathsf{qSCom}_{pk}()$ produces a soft commitment $C$ and an auxiliary information $\mathsf{aux}$. A soft commitment string $C$ is created to no specific sequence of messages.

$\mathsf{qSOpen}_{pk}(m, i, \mathsf{flag}, \mathsf{aux})$ produces a soft decommitment $\tau$ (also known as *"tease"*) to a message $m$ at position $i$. The parameter $\mathsf{flag} \in \{\mathbb{H}, \mathbb{S}\}$ indicates if $\tau$ corresponds to either a hard commitment $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$ or to a soft commitment $(C, \mathsf{aux}) = \mathsf{qSCom}_{pk}()$. The algorithm returns an error message if $C$ is an hard commitment and $m \neq m_i$.

$\mathsf{qSVer}_{pk}(m, i, C, \tau)$ checks if $\tau$ is a valid decommitment for $C$ to $m$ of index $i$. If it outputs 1 and $\tau$ corresponds to a hard commitment $C$ to $(m_1, \cdots, m_q)$, then $C$ could be hard-opened to $(m, i)$, or rather $m = m_i$.

$\mathsf{qFake}_{pk,tk}()$ takes as input the trapdoor $tk$ and produces a $q$-fake commitment $C$. $C$ is not bound to any sequence $(m_1, \cdots, m_q)$. It also returns an auxiliary information $\mathsf{aux}$.

$\mathsf{qHEquiv}_{pk,tk}(m_1, \cdots, m_q, i, \mathsf{aux})$ The non-adaptive hard equivocation algorithm generates a hard decommitment $\pi$ for $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$ to the $i$-th message of $(m_1, \cdots, m_q)$. The algorithm is non adaptive in the sense that, for a given $C$, the sequence $(m_1, \cdots, m_q)$ has to be determined once and for all, before $\mathsf{qHEquiv}$ is executed. A $q$-fake commitment is very similar to a soft commitment with the additional property that it can be hard-opened.

$\mathsf{qSEquiv}_{pk,tk}(m, i, \mathsf{aux})$ generates a soft decommitment $\tau$ to $m$ of position $i$ using the auxiliary information produced by the $\mathsf{qFake}$ algorithm.

The correctness requirements for trapdoor $q$-Mercurial commitments are essentially the same as those for "traditional" commitment schemes. In particular we require that $\forall (m_1, \cdots, m_q) \in \mathcal{M}^q$, the following statements are false only with negligible probability.

1. if $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$:
$$\mathsf{qHVer}_{pk}(m_i, i, C, \mathsf{qHOpen}_{pk}(m_i, i, \mathsf{aux})) = 1 \quad \forall i = 1 \ldots q$$
2. If $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$
$$\mathsf{qSVer}_{pk}(m_i, i, C, \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathsf{aux})) = 1 \quad \forall i = 1 \ldots q$$
3. If $(C, \mathsf{aux}) = \mathsf{qSCom}_{pk}()$
$$\mathsf{qSVer}_{pk}(m_i, i, C, \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{S}, \mathsf{aux})) = 1 \quad \forall i = 1 \ldots q$$
4. If $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$
$$\mathsf{qHVer}_{pk}(m_i, i, C, \mathsf{qHEquiv}_{pk,tk}(m_1, \cdots, m_q, i, \mathsf{aux})) = 1$$
$$\mathsf{qSVer}_{pk}(m_i, i, C, \mathsf{qSEquiv}_{pk,tk}(m_i, i, \mathsf{aux})) = 1 \quad \forall i = 1 \ldots q$$

**Security** The security properties for a trapdoor $q$-mercurial commitment scheme are as follows:

– ***q*-Mercurial binding**. Having knowledge of $pk$ it is computationally infeasible for an algorithm $\mathcal{A}$ to come up with $C, m, i, \pi, m', \pi'$ such that either one of the following cases holds:

- $\pi$ is a valid hard decommitment for $C$ to $(m, i)$ and $\pi'$ is a valid hard decommitment for $C$ to $(m', i)$, with $m \neq m'$. We call such case a "hard collision".
- $\pi$ is a valid hard decommitment for $C$ to $(m, i)$ and $\pi'$ is a valid soft decommitment for $C$ to $(m', i)$, with $m \neq m'$. We call such case a "soft collision".

– $q$**-Mercurial hiding**. There exists no PPT adversary $\mathcal{A}$ that, knowing $pk$, can find a tuple $(m_1, \cdots, m_q) \in \mathcal{M}^q$ and an index $i$ for which it can distinguish $(C, \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathsf{aux}))$ from $(C', \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{S}, \mathsf{aux}'))$, where $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$ and $(C', \mathsf{aux}') = \mathsf{qSCom}_{pk}()$.

– **Equivocations**. There exists no PPT adversary $\mathcal{A}$ that, knowing $pk$ and the trapdoor $tk$, can win any of the following games with non-negligible probability. In such games $\mathcal{A}$ should be able to tell apart the "real" world from the corresponding "ideal" one. The games are formalized in terms of a challenger that flips a binary coin $b \in \{0, 1\}$. If $b = 0$ it gives to $\mathcal{A}$ a real commitment/decommitment tuple; if $b = 1$ it gives to $\mathcal{A}$ an ideal tuple produced using the fake algorithms.
In the q-HHEquivocation and the q-HSEquivocation games below, $\mathcal{A}$ chooses $(m_1, \cdots, m_q) \in \mathcal{M}^q$ and receives a commitment string $C$. Then $\mathcal{A}$ gives an index $i \in \{1, \cdots, q\}$ to the challenger and finally it receives a hard decommitment $\pi$.
  - **q-HHEquivocation.** If $b = 0$ the challenger hands to $\mathcal{A}$ the value $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$. $\mathcal{A}$ gives $i$ to the challenger and gets back $\pi = \mathsf{qHOpen}_{pk}(m_i, i, \mathsf{aux})$. Otherwise the challenger computes $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$, $\pi = \mathsf{qHEquiv}_{pk,tk}(m_1, \cdots, m_q, i, \mathsf{aux})$.
  - **q-HSEquivocation.** The challenger computes $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$, $\pi = \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathsf{aux})$ in the case $b = 0$ or $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}(), \pi = \mathsf{qSEquiv}_{pk,tk}(m_i, i, \mathsf{aux})$ if $b = 1$.
  - **q-SSEquivocation.** If $b = 0$ the challenger generates $(C, \mathsf{aux}) = \mathsf{qSCom}_{pk}()$ and gives $C$ to $\mathcal{A}$. Next, $\mathcal{A}$ chooses $m \in \mathcal{M}$ and an index $i \in \{1, \cdots, q\}$, it gives $(m, i)$ to the challenger and receives back $\mathsf{qSOpen}_{pk}(m, i, \mathbb{S}, \mathsf{aux})$. If $b = 1$, $\mathcal{A}$ first gets $\mathsf{qFake}_{pk,tk}()$, then it chooses $m \in \mathcal{M}, i \in \{1, \cdots, q\}$, gives $(m, i)$ to the challenger and gets back $\mathsf{qSEquiv}_{pk,tk}(m, i, \mathsf{aux})$.

  At some point $\mathcal{A}$ outputs $b'$ as its guess for $b$ and wins if $b' = b$.
  As for the case of trapdoor mercurial commitments (see [4]) it is easy to see that the $q$-mercurial hiding is implied by the $q$-HSEquivocation and $q$-SSEquivocation.

### 2.2 Zero-Knowledge Sets

Zero knowledge sets [17] allows one to commit to some secret set $S$ and then to, non interactively, produce proofs of the form $x \in S$ or $x \notin S$. This is done without revealing any further information (i.e. that cannot be deduced by the statements above) about $S$, not even its size. Following the approach of [17], here we focus on the more general notion of zero-knowledge elementary databases (EDB), since the notion of zero-knowledge sets is a special case of zero-knowledge EDBs (see [17] for more details about this). Let $[D]$ be the set of keys associated to a database $D$. We assume that $[D]$ is a proper subset of $\{0,1\}^*$. If $x \in [D]$, we denote with $y = D(x)$ its associated value in the database $D$. If $x \notin [D]$ we let $D(x) = \perp$. An EDB system is formally defined by a triple of algorithms $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$:

- $\mathsf{P}_1$, the *committer* algorithm, takes in input a database $D$ and the common reference string $CRS$ and outputs a public key $ZPK$ and a secret key $ZSK$.
- On input the common reference string $CRS$, the secret key $ZSK$ and an element $x$, the *prover* algorithm $\mathsf{P}_2$ produces a proof $\pi_x$ of either $D(x) = y$ or $D(x) = \perp$.
- The third algorithm is the *verifier* $\mathsf{V}(CRS, ZPK, x, \pi_x)$. It outputs $y$ if $D(x) = y$, *out* if $D(x) = \perp$ or $\perp$ if the proof $\pi_x$ is not valid.

The formal definition of *zero-knowledge EDB* is given in [17][8].

## 3 Zero Knowledge EDB from trapdoor q-Mercurial Commitments

In this section we describe a construction of zero-knowledge EDB, from trapdoor $q$-mercurial commitments (defined in section 2.1), trapdoor mercurial commitments [5, 4] and collision resistant hash functions. The construction is very simple, as it generalizes easily from the original [17, 5] constructions. Still, it plays an important role in our quest for efficient zero knowledge sets, as it allows us to concentrate solely on the problem of realizing efficient qTMCs.

**Intuitive construction** Assume we want to commit to a database $D$ with keys of length $k$. We associate each key $x$ to a leaf in a $q$-ary tree of height $k$. Thus $x$ can be viewed as a number representing the labeling of the leaf in $q$-ary encoding (see the example in Figure 1). Since the number of all possible keys is $q^k$, to make the committing phase efficient (i.e. polynomial in $k$) the tree is pruned by cutting those subtrees containing only keys of elements not in the database. The roots of such subtrees are kept in the tree (we call them the "frontier"). The internal nodes in the frontier are "filled" with soft commitments. The remaining nodes are filled as follows. Each leaf contains an hard commitment (computed using the standard trapdoor mercurial commitment scheme) of a value $n_{H(x)}$ related to $D(x)$[9]. Each internal node contains the hard $q$-commitment to the hashes of the values contained in its $q$ sons. The $q$-commitment contained in the root of the tree is the public key of the zero-knowledge EDB.

[8] We point out here that we will prove our construction secure with respect to a slightly different definition (with respect to the one given in [17]) in which the completeness requirement is relaxed to allow a negligible probability of error.

[9] More precisely $n_{H(x)}$ is the hash of $D(x)$ if $x$ is in the database and 0 otherwise.

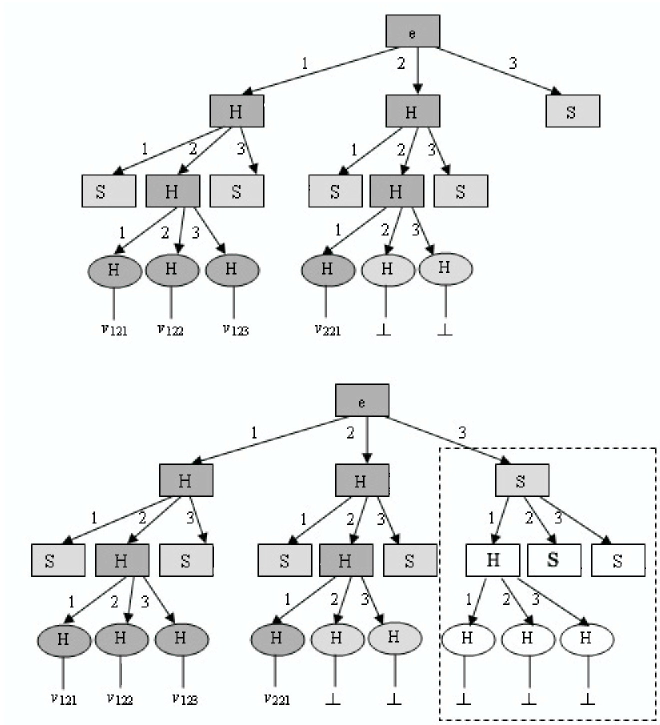


**Fig. 1. A 3-ary tree of height 3 before and after a query to the database key 311.** Each node of the tree contains a mercurial commitment: the label $H$ is for hard commitments, $S$ for the soft ones. Moreover the squares represent $q$-commitments, while the circles represent standard commitments. If the set of database keys is $S = \{121, 122, 123, 221\}$, the darker nodes are those belonging to a path from the root to an element in the set. The light shaded nodes are the frontier.

When the prover $P$ is asked for a proof of an element $x \in D$ (for instance such that $D(x) = y$), it proceeds as follows. It exhibits hard openings for the commitments contained in the nodes in the path from the root to the leaf $x$. More precisely, for each level of the tree, it opens the hard $q$-commitment with respect to the position determined by the $q$-ary encoding of $x$ for that level. Queries corresponding to keys $x$ such that $D(x) = \perp$ are answered as follows. First, the prover generates the possibly missing portion of the subtree containing $x$. Next, it soft opens all the commitments contained in the nodes in the path from $x$ to the root. The soft commitments stored in the frontier nodes are then teased to the values contained in its newly generated children.

It is easy to see that the completeness property follows from the completeness of the two commitment schemes used. Similarly, the binding properties of the two commitment schemes, together with the collision resistance of the underlying hash function, guarantees that (1) no hard commitment can be opened to two different values, and (2) no hard commitment can be opened to a value and then teased to a different one.

Finally the zero-knowledge property follows from the fact that both the two commitments schemes are hiding and equivocable (the fake commitments and fake openings produced by the simulator are indistinguishable from the commitments and openings produced from a real prover).

A detailed description of the construction sketched above, together with a complete security proof, is given in the full version of this paper.

## 4 Trapdoor $q$-Mercurial Commitment based on SDH

In this section we show an efficient construction of trapdoor $q$-mercurial commitments $\mathcal{QC}$.

Our construction relies on the Strong Diffie-Hellman assumption (SDH for short), introduced by Boneh and Boyen in [3]. 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:

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

If we suppose that $\mathcal{G}(1^k)$ is a bilinear group generator which takes in input a security parameter $k$, then (asymptotically) the SDH assumption holds for $\mathcal{G}$ if the probability above is negligible in $k$, for any $q$ polynomial in $k$ (see [3] for the formal definition).

The SDH assumption obviously implies the discrete logarithm assumption (i.e. if the former holds, so has to do the latter). A reduction in the other direction, however, is not known. Recently, however, Cheon [6] proved that, for many primes $p$, the $q$-Strong Diffie Hellman problem has computational complexity reduced by $O(\sqrt{q})$ with respect to that of the discrete logarithm problem (in the same group).

The new scheme. Now we describe our proposed trapdoor $q$-Mercurial Commitment scheme, in terms of its component algorithms ($\mathsf{qKeyGen}$, $\mathsf{qHCom}$, $\mathsf{qHOpen}$, $\mathsf{qHVer}$, $\mathsf{qSCom}$, $\mathsf{qSOpen}$, $\mathsf{qSVer}$, $\mathsf{qFake}$, $\mathsf{qHEquiv}$, $\mathsf{qSEquiv}$), as described in section 2.1.

The technical construction of the proposed scheme builds upon the simulator described in the security proof of the weak signature scheme given in [3].

In what follows $\mathcal{H}$ denotes a family of collision resistant hash functions whose range is $\mathbb{Z}_p$.

$\mathsf{qKeyGen}(1^k, q)$ The key generation algorithm runs a bilinear group generator $\mathcal{G}(1^k)$ for which the SDH assumption holds [3] to get back the description of groups $G_1, G_2, G_T$ and a bilinear map $e : G_1 \times G_2 \to G_T$. Such groups share the same prime order $p$.
The description of the groups contains the group generators: $g_1 \in G_1$, $g_2 \in G_2$. The algorithm proceeds by picking a random integer $x \leftarrow \mathbb{Z}_p^*$ and it sets

$A_1 = g_1^x, \cdots, A_q = g_1^{x^q}, h = g_2^x$. Next, it chooses a collision resistant hash function $H$ from $\mathcal{H}$.

The public key is set as $PK = (g_1, A_1, \cdots, A_q, g_2, h, H)$, while the trapdoor is $TK = x$.

$\mathsf{qHCom}_{PK}(m_1, \cdots, m_q)$ . The hard commitment algorithm randomly selects $\alpha, w \leftarrow \mathbb{Z}_p^*$ and computes $C_i = H(i||m_i), \forall i = 1, \cdots, q$ (the symbol $||$ denotes concatenation). Next, it defines the polynomial

$$f(z) = \prod_{i=1}^{q}(z + C_i) = \sum_{i=0}^{q}(\beta_i z^i)$$

and sets $g_1' = (\prod_{i=0}^{q} A_i^{\alpha^i \beta_i})^w = g_1^{f(\alpha x)w}$ and $g_2' = h^\alpha$. In the unlucky case that either $g_1' = 1$ or $g_2' = 1$, then one simply retries with another random $\alpha$.

Thus, letting $\gamma = \alpha x$, we have $g_1' = g_1^{f(\gamma)w}$ and $g_2' = h^\alpha = g_2^\gamma$.

The commitment is $(g_1', g_2')$. The auxiliary information is $\mathsf{aux} = (\alpha, w, m_1, \cdots, m_q)$.

$\mathsf{qHOpen}_{PK}(m, j, \mathsf{aux})$ outputs $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$.

$\mathsf{qHVer}_{PK}(m, j, C, \pi)$ computes the $q - 1$ terms $C_i = H(i||m_i) \forall m_i \in \pi$ and $C_j = H(j||m)$. Next, it defines the polynomial $f(z) = \prod_{i=1}^{q}(z + C_i)$ and computes the $\beta_i$ coefficients as above.

Checks if $g_1' = (\prod_{i=0}^{q} A_i^{\alpha^i \beta_i})^w$ and $g_2' = h^\alpha$. If both tests succeed, it outputs 1.

$\mathsf{qSCom}_{PK}()$ picks $\alpha', y \leftarrow \mathbb{Z}_p^*$ at random, sets

$$g_1' = g_1^{\alpha'}, \;\; g_2' = g_2^y$$

and outputs $(g_1', g_2')$ and $\mathsf{aux} = (\alpha', y)$.

$\mathsf{qSOpen}_{PK}(m, j, \mathsf{flag}, \mathsf{aux})$ If $\mathsf{flag} = \mathbb{H}$ the algorithm computes $C_i = H(i||m_i)$, $\forall i = 1, \cdots, j-1, j+1, \cdots, q$, $C_j = H(j||m)$, it sets

$$f_j(z) = \frac{f(z)}{(z + C_j)} = \prod_{i=1 \wedge i \neq j}^{q}(z + C_i) = \sum_{i=0}^{q-1} \delta_i z^i$$

Next, it computes $\sigma_j = (\prod_{i=0}^{q-1} A_i^{\delta_i \alpha^i})^w = g_1^{\frac{f(\gamma)w}{\gamma + C_j}} = (g_1')^{\frac{1}{\gamma + C_j}}$. The output is $\sigma_j$.

If $\mathsf{flag} = \mathbb{S}$ the algorithm computes $C_j = H(j||m)$ and outputs $\sigma_j = (g_1')^{\frac{1}{y + C_j}}$.

$\mathsf{qSVer}_{PK}(m, j, C, \tau)$ The soft verification algorithm takes in input a message $m$ and an index $j \in \{1, \cdots, q\}$. It computes $C_j = H(j||m_j)$, and checks if $e(\sigma_j, g_2' g_2^{C_j}) = e(g_1', g_2)$. If this is the case, it outputs 1.

$\mathsf{qFake}_{PK,TK}()$ The fake commitment algorithm is the same as $\mathsf{qSCom}$.

$\mathsf{qHEquiv}_{PK,TK}(m_1, \cdots, m_q, j, \mathsf{aux})$ The non-adaptive hard equivocation algorithm uses the trapdoor key $TK$ to hard open a fake commitment (which is originally a commitment to nothing). It computes $C_i = H(i||m_i), \forall i = 1, \cdots, q$ and constructs the polynomial

$$f(z) = \prod_{i=1}^{q}(z + C_i) = \sum_{i=0}^{q} \beta_i z^i.$$

It sets $\alpha = \frac{y}{x}, w = \frac{\alpha'}{f(y)}$ and outputs $\pi = \{\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q\}$.

$\mathsf{qSEquiv}_{PK,TK}(m, j, \mathsf{aux})$ The soft equivocation algorithm is the same as $\mathsf{qSOpen}$.

### 4.1 Properties of the scheme

First notice that our commitment scheme is "proper" in the sense of [4]. Recall that a mercurial commitment scheme is said to be "proper" if the soft decommitment is a proper subset of the hard decommitment. In our scheme, a soft decommitment is implicitly contained in a hard one. Indeed, given a hard opening $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$ to a message $m$ at position $j$ and the public key $PK$, we are able to compute a valid soft decommitment $\sigma_j$ to the message $m$ of index $j$.

The correctness of the scheme can be easily verified by inspection. With the next theorem we show that the remaining properties of $\mathsf{qTMC}$ are realized as well.

**Theorem 1.** *Assuming that the Strong Diffie-Hellmann holds for $\mathcal{G}$ and $\mathcal{H}$ is a family of collision resistant hash functions, $\mathcal{QC}$ is a trapdoor q-mercurial commitment scheme.*

*Proof (Theorem 1).* To prove the theorem we need to make sure that the proposed scheme is binding and hiding, in the sense discussed in section 2.1. We prove each property separately.

**$q$-mercurial binding**. To prove the property we need to make sure that neither hard collisions nor soft ones are possible. We prove that it is infeasible to find any of such collisions under the Strong Diffie Hellmann assumption (SDH) for the bilinear group generator $\mathcal{G}$ [3] and the collision resistance of the hash function $H$.

Let us first consider soft collisions. Next we describe how to adapt the same proof for the case of hard collisions.

Soft collisions. Assume there exists an adversary $\mathcal{A}^{\mathbb{S}}$ that with non-negligible probability $\epsilon$ can find a soft collision. We show how to build a simulator $\mathcal{B}^{\mathbb{S}}$ that uses $\mathcal{A}^{\mathbb{S}}$ to solve the $q$-SDH problem, or to break the collision resistance of $H$, with probability at least $\epsilon/2$.

$\mathcal{B}^{\mathbb{S}}$ receives in input from its challenger a $(q+3)$-tuple $(g_1, g_1^x, \cdots, g_1^{x^q}, g_2, g_2^x)$ and the description of a hash function $H$. The simulator runs $\mathcal{A}^{\mathbb{S}}$ on input such values as the public key of the $q$-mercurial commitment scheme. Then with probability $\epsilon$ the adversary outputs $(C, m, j, \pi, m', \tau)$ such that: $C = (g_1', g_2')$ is a commitment, $m \neq m'$, $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$ is a valid hard opening for $C$ to the message $m$ at position $j$ and $\tau = (\sigma_j)$ is a valid soft opening for $C$ to $m'$ of index $j$. We distinguish two cases:

1. $m \neq m'$ and $C_j = H(j||m) = H(j||m') = C'_j$;
2. $m \neq m'$ and $C_j \neq C'_j$.

At least one of these cases occurs with probability at least $\epsilon/2$. In the first case the simulator immediately has a collision for $H$. In case 2 we show how to solve the $q$-SDH problem.

Since $\mathsf{qSVer}_{PK}(m', j, C, \tau) = 1$ we have that $e(\sigma_j, g'_2 g_2^{C'_j}) = e(g'_1, g_2)$. Moreover, the correct verification of $\pi$ implies that $g'_2 = h^\alpha = g_2^\gamma$ thus $\sigma_j = (g'_1)^{\frac{1}{\gamma + C'_j}}$.

Using long division we can write the $q$-degree polynomial $f$ as $f(z) = \eta(z)(z + C'_j) + \eta_{-1}$ where $\eta(z) = \sum_{i=0}^{q-1} \eta_i z^i$ is a polynomial of degree $q-1$ and $\eta_{-1} \in \mathbb{Z}_p$. Thus we can write $\sigma_j = (g_1^{\eta(\gamma)} g_1^{\frac{\eta_{-1}}{\gamma + C'_j}})^w$. Hence first $\mathcal{B}^{\mathbb{S}}$ computes:

$$\delta = (\sigma_j^{1/w} \cdot \prod_{i=0}^{q-1} A_i^{-\eta_i \alpha^i})^{1/\eta_{-1}} = (g_1^{\eta(\gamma)} g_1^{\frac{\eta_{-1}}{\gamma + C'_j}} g_1^{-\eta(\gamma)})^{1/\eta_{-1}} = g_1^{\frac{1}{\gamma + C'_j}}.$$

Finally it computes $\delta^* = \delta^\alpha = g_1^{\frac{\alpha}{\alpha x + C'_j}} = g_1^{\frac{1}{x + C'_j/\alpha}}$ and $C^* = C'_j/\alpha$. The simulator gives $(\delta^*, C^*)$ to its challenger. It is easy to see that such pair breaks the $q$-SDH assumption. Thus with non-negligible advantage $\epsilon/2$ $\mathcal{B}^{\mathbb{S}}$ can break either the $q$-SDH assumption or the collision resistance of $H$.

Hard collisions. Let us now assume there exists an adversary $\mathcal{A}^{\mathbb{H}}$ that, given the public key of a $q$-mercurial commitment scheme, can find a hard collision with non-negligible probability $\epsilon$. Then we construct a simulator $\mathcal{B}^{\mathbb{H}}$ that either solves the $q$-SDH problem or breaks the collision resistance of $H$ with probability at least $\epsilon/2$. The simulator $\mathcal{B}^{\mathbb{H}}$ is similar to the one described above. The difference is that $\mathcal{A}^{\mathbb{H}}$ outputs: $(C, m, j, \pi, m', \pi')$ such that: $C = (g'_1, g'_2)$ is a commitment, $m \neq m'$ are two different messages, $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$ is a valid hard opening for $C$ to $m$ of index $j$ and $\pi' = (\alpha', w', m'_1, \cdots, m'_{j-1}, m'_{j+1}, \cdots, m'_q)$ is a valid hard opening for $C$ to $m'$ of index $j$. Again we consider two cases:

1. $m \neq m'$ and $C_j = H(j||m) = H(j||m') = C'_j$,
2. $m \neq m'$ and $C_j \neq C'_j$.

Case 1 is the same as before. In case 2, $\mathcal{B}^{\mathbb{H}}$ solves the $q$-SDH problem as follows. Since $\mathsf{qHVer}_{PK}(m, j, C, \pi) = 1$ and $\mathsf{qHVer}_{PK}(m', j, C, \pi') = 1$, it must be the case that $\alpha = \alpha'$ ($\alpha \neq \alpha'$, would lead to two different $g'_2$ $h^\alpha$ and $h^{\alpha'}$). Moreover, since the commitment scheme is proper from the valid hard opening $\pi' = (\alpha', w', m'_1, \cdots, m'_{j-1}, m'_{j+1}, \cdots, m'_q)$ for $m'_j$ we can "extract" a valid soft opening for $m'_j$. Thus, using exactly the same argument described above, we break the SDH assumption.

**Hiding and Equivocation**. First notice that, since our scheme is proper, it suffices to check only $q$-HHEquivocation and $q$-SSEquivocation hold. In both cases we show that it is infeasible for an adversary to distinguish between a real commitment/decommitment tuple from a fake/equivocation one.

In the $q$-HHEquivocation game the adversary is asked to tell apart

$$\{(g_1^{f(\gamma)w}, g_2^{\alpha x}), (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)\}$$

from

$$\{(g_1^{\alpha'}, g_2^{y}), (\alpha = \frac{y}{x}, w = \frac{\alpha'}{f(\gamma)}, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)\}$$

In both cases $\alpha, w$ are uniformly random in $\mathbb{Z}_p^*$. This is because, in the first tuple, they are chosen uniformly and at random, while in the second tuple they are distributed, respectively, as $y$ and $\alpha'$, which were chosen uniformly and at random in $\mathbb{Z}_p^*$.

Thus the two distributions are indistinguishable.

The proof of indistinguishability for the $q$-SSEquivocation is trivial. Indeed, it is easy to see that the elements in the two distributions

$$\{(g_1^{\alpha'}, g_2^{y}), \sigma_i = (g_1')^{\frac{1}{\gamma + C_i}}\}$$

$$\{(g_1^{\alpha'}, g_2^{y}), \sigma_i = (g_1')^{\frac{1}{\gamma + C_i}}\}$$

are distributed in exactly the same manner.

## 5 Efficiency considerations

In the previous section we proposed a trapdoor $q$-mercurial commitment scheme $\mathcal{QC}$ based on the Strong Diffie-Hellmann assumption. In order to build efficient zero knowledge EDB, we also use a trapdoor mercurial commitment scheme $\mathcal{C}$ based on the Discrete Logarithm constructions given in [17, 5]. For our convenience we consider an implementation of the scheme that allows us to use some of the parameter already in use for the $\mathsf{qTMC}$ scheme. In particular, we use $g_1, A_1 \in G_1$ from the public key of $\mathcal{QC}$ as the public key for $\mathcal{C}$.

Combining the two schemes as described in section 3, we obtain an implementation of zero-knowledge EDB (based on the SDH problem) that allows for proofs that are significantly shorter than those produced by previous proposals.

Below we compare our proposal with the most efficient (in terms of space) implementation known so far, namely the one by Micali *et al.* [17] (MRK from now on, for short), when implemented over elliptic curves with short representation.

We measure efficiency in terms of the space taken by each proof. For both schemes, we assume that the universe $\mathcal{U}$ has size $|\mathcal{U}| = 2^k = q^h$ and, that $q = 2^{k'}$, for simplicity.

**Groups used in the comparisons** Following [10] we fix a security parameter $\ell = 256$ to achieve $k = 128$ bits of security. Specifically $G_1$ is realized as a subgroup of points on an elliptic curve $E$ over a finite field $\mathbb{F}_p$ of size $p$, where $p$ is an $\ell$ bits prime. If $e$ is a parameter called *embedding degree*, $G_2$ is a subgroup of $E(\mathbb{F}_{p^e})$ and $G_T \subset E(\mathbb{F}_{p^e}^*)$. In particular we consider elliptic curves with embedding degree $e = 12$ and CM discriminant $D = -3$. As suggested in [10], for the case of *Type 3 groups* (see [10] for details), such parameters enable to obtain elements of $G_2$ that have size twice the size of elements of $G_1$.

| $q$ | Membership | Non-membership |
|---|---|---|
| 2 | 773 | 516 |
| 4 | 517 | 260 |
| 8 | 517 | 174.7 |
| 16 | 645 | 132 |
| 32 | 926.6 | 106.4 |
| 64 | 1455.7 | 89.3 |
| 128 | 2418.7 | 77.1 |
| 256 | 4165 | 68 |

**Table 1.** Space required by proofs, in our scheme

**Bandwidth** A proof of membership in our scheme contains $h(q + 4) + 5$ elements[10]. A proof of non-membership $4h + 4$. In MRK's scheme a proof of membership requires $6k + 5$ elements, while a proof of non-membership needs $5k + 4$ elements. In both cases all the elements have size $\ell$, but, for our scheme, we let $q$ vary. For such a choice of parameters we obtain the following results.

The scheme of Micali *et al.* requires 773 elements for proofs of membership and 644 for proofs of non-membership. Results for our scheme are summarized in Table 1.

Notice that our scheme produces proofs of non-membership, that are always much shorter than the corresponding MRK proofs. The space required by our proofs of membership, on the other hand, compares favorably to MRK scheme only until $q \leq 16$, it gets slightly worse for $q = 32$, and much worse for larger values of $q$. Thus, the choice of $q = 8$ leads to proofs of membership that are (approximately) 33% shorter, and to proofs of non membership that are almost 73% shorter than MRK!

Notice that such a choice of $q$ (i.e. $q = 8$) keeps the scheme practical also in terms of length of the common reference string. Notice also that, according to our present knowledge of the SDH problem, it seems reasonable to consider the same security parameter for our scheme and for the MRK implementation. This is because Cheon [6] attack requires $q$ to be an upper-bound to a factor of either $p - 1$ or $p + 1$ in order to be effective. If one sets $q = 8$, as suggested in the table above, this would imply that one should increase the key size of at most 2 bits in the worst case. Thus using the same security parameter for both ours and MRK seems to be reasonable for all practical purposes.

## 6 Conclusions

In this paper we introduced and implemented the notion of trapdoor $q$ mercurial commitments. Our construction can be used to construct zero knowledge sets

[10] We assume each element has size $\ell$. This is because, the size of each element in $G_2$ is twice that of an element in $G_1$. Thus whenever an element in $G_2$ is considered, this counts as two elements in $G_1$.

that allow for proofs that are much shorter than those obtained by previous work. It would be interesting to investigate if it is possible to come up with an even more efficient implementation of the new primitive. In particular, it would be very interesting to construct a $\mathsf{qTMC}$ that allows for openings whose length is independent of $q$.

## References


1. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *ACM Conference on Computer and Communications Security, 1993*
2. M. Blum, A. De Santis, S. Micali and P. Persiano. Non Interactive Zero Knowledge. *SIAM Journal on Computing*, 20(6), 1991.
3. D. Boneh and X. Boyen. Short Signatures Without Random Oracles. *Advances in Cryptology – proceedings of EUROCRYPT 2004, LNCS 3027*
4. D. Catalano, Y. Dodis and I. Visconti. Mercurial Commitments: Minimal Assumptions and Efficient Constructions. *Theory of Cryptography Conference – TCC'06*
5. M. Chase, A. Healy, A. Lysyanskaya, T. Malkin and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. *Advances in Cryptology – proceedings of EUROCRYPT 2005, LNCS 3494*
6. J. Hee Cheon Security Analysis of the Strong Diffie-Hellman Problem. *Advances in Cryptology – proceedings of EUROCRYPT 2006, LNCS 4004*
7. R. Cramer and I. Damgård. New Generation of Secure and Practical RSA-based signatures. *Proc. of Crypto '96* LNCS no. 1109, pages 173-185.
8. I. Damgård. Collision free hash functions and public key signature schemes. *Proc. of Eurocrypt '87* LNCS no. 304, pages 203-216.
9. C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. *J. of Cryptology* 11(3) 1998, pages 187-208.
10. S.D. Galbraith, K.G. Paterson and N.P. Smart Pairings for Cryptographers *Cryptology ePrint Archive, Report 2006/165, 2006.* `http://eprint.iacr.org/`.
11. R. Gennaro and S. Micali. Independent Zero-Knowledge Sets. *In proceedings of ICALP '06, 2006*
12. S. Goldwasser and R.Ostrovsky Invariant Signatures and Non Interactive Zero Knowledge proofs are equivalent *Advances in Cryptology – proceedings of CRYPTO 92*
13. C.H. Lim. Efficient Multi-Exponentiation and Application to Batch Verification of Digital Signatures. *Unpublished manuscript, August 2000*
14. C.H. Lim. and P.J. Lee More Flexible Exponentiation with Precomputation. *Advances in Cryptology – proceedings of CRYPTO 94, LNCS 839*
15. M. Liskov. Updatable zero-knowledge databases. *Advances in Cryptology – proceedings of ASIACRYPT '05, 2005*
16. R. Merkle. A Digital Signature based on a Conventional Encryption Function. *Advances in Cryptology–Crypto'87.* LNCS, vol.293, pp. 369–378, Springer–Verlag, 1988.
17. S. Micali, M. Rabin and J.K. Kilian. Zero-Knowledge Sets. *In proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science – FOCS'03*
18. S. Micali, M. Rabin and S. Vadhan. Verifiable Random Functions. *In proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science – FOCS'99*

19. R. Ostrovsky, C. Rackoff and A. Smith Efficient consistency proof on a committed database. In *Proc. of ICALP* 2004.
20. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Crypto'91*, pp.129-140, Lecture Notes in Computer Science vol.576, Springer-Verlag, 1992.
21. M. Prabhakaran and R. Xue. Statistically Hiding Sets *Cryptology ePrint Archive, Report 2007/349, 2007.* http://eprint.iacr.org/.