Concise Mercurial Vector Commitments and Independent Zero-Knowledge Sets with Short Proofs

Benoît Libert[1] * and Moti Yung[2]

[1] Université catholique de Louvain, Crypto Group (Belgium)
[2] Google Inc. and Columbia University (USA)

Abstract. Introduced by Micali, Rabin and Kilian (MRK), the basic primitive of zero-knowledge sets (ZKS) allows a prover to commit to a secret set S so as to be able to prove statements such as $x \in S$ or $x \notin S$. Chase *et al.* showed that ZKS protocols are underlain by a cryptographic primitive termed *mercurial commitment*. A (trapdoor) mercurial commitment has two commitment procedures. At committing time, the committer can choose not to commit to a specific message and rather generate a dummy value which it will be able to softly open to any message without being able to completely open it. Hard commitments, on the other hand, can be hardly or softly opened to only one specific message. At Eurocrypt 2008, Catalano, Fiore and Messina (CFM) introduced an extension called trapdoor q-mercurial commitment (qTMC), which allows committing to a vector of q messages. These qTMC schemes are interesting since their openings w.r.t. specific vector positions can be short (ideally, the opening length should not depend on q), which provides zero-knowledge sets with much shorter proofs when such a commitment is combined with a Merkle tree of arity q. The CFM construction notably features short proofs of *non-membership* as it makes use of a qTMC scheme with short soft openings. A problem left open is that hard openings still have size $O(q)$, which prevents proofs of membership from being as compact as those of non-membership. In this paper, we solve this open problem and describe a new qTMC scheme where hard and short position-wise openings, both, have *constant size*. We then show how our scheme is amenable to constructing independent zero-knowledge sets (i.e., ZKS's that prevent adversaries from correlating their set to the sets of honest provers, as defined by Gennaro and Micali). Our solution retains the short proof property for this important primitive as well.

Keywords. Zero-knowledge databases, mercurial commitments, efficiency, independence.

1 Introduction

Introduced by Micali, Rabin and Kilian [21], zero-knowledge sets (ZKS) are fundamental secure data structures which allow a prover P to commit to a finite set

* This author acknowledges the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.) for their support and the BCRYPT Interuniversity Attraction Pole.

S in such a way that, later on, he will be able to efficiently (and non-interactively) prove statements of the form $x \in S$ or $x \notin S$ without revealing anything else on S, not even its size. Of course, the prover should not be able to cheat and prove different statements about an element x. The more general notion of zero-knowledge elementary databases (ZK-EDB) generalizes zero-knowledge sets in that each element x has an associated value $D(x)$ in the committed database.

In [21], Micali *et al.* described a beautiful construction of ZK-EDB based on the discrete logarithm assumption. The MRK scheme relies on the shared random string model (where a random string chosen by some trusted entity is made available to all parties) and suitably uses an extension of Pedersen's trapdoor commitment [23]. In 2005, Chase *et al.* [10] gave general constructions of zero-knowledge databases and formalized a primitive named *mercurial commitment* which they proved to give rise to ZK-EDB protocols. The MRK construction turned out to be a particular instance of a general design combining mercurial commitments with a Merkle tree [20], where each internal node contains a mercurial commitment to its two children.

Informally speaking, mercurial commitments are commitments where the binding property is slightly relaxed in that the committer is allowed to *softly* open a commitment and say "if the commitment can be opened at all, then it opens to that message". Upon committing, the sender has to decide whether the commitment will be a hard commitment, that can be hard/soft-opened to only one message, or a soft one that can be soft-opened to any arbitrary message without committing the sender to a specific one. Unlike soft commitments that cannot be hard-opened, hard commitments can be opened either in the soft or the hard manner but soft openings can never contradict hard ones. In addition, hard and soft commitments should be computationally indistinguishable.

Related Work. Promptly after the work of Micali, Rabin and Kilian, Ostrovsky, Rackoff and Smith [22] described protocols for generalized queries (beyond membership/non-membership) for committed databases and also show how to add privacy to their schemes. Liskov [18] also extended the construction of Chase *et al.* [10] to obtain updatable zero-knowledge databases in the random oracle model. Subsequently, Catalano, Dodis and Visconti [8] gave simplified security definitions for (trapdoor) mercurial commitments and notably showed how to construct them out of one-way functions in the shared random string model.

In order to extend the properties of non-malleable commitments to zero-knowledge databases, Gennaro and Micali [15] formalized the notion of *independent* ZK-EDBs. Informally, this notion prevents adversaries from correlating their committed databases to those produced by honest provers.

More recently, Prabhakaran and Xue [24] defined the related notion of statistically hiding sets that requires the hiding property of zero-knowledge sets to be preserved against unbounded verifiers. At the same time, their notion of zero-knowledge was relaxed to permit unbounded simulators.

At Eurocrypt 2008, Catalano, Fiore and Messina [9] addressed the problem of compressing proofs in ZK-EDB schemes and gave significant improvements.

Our Contribution. The original construction of zero-knowledge database [21, 10] considers a binary Merkle tree of height $O(\lambda)$, where λ is the security parameter (in such a way that the upper bound on the database size is exponential in λ and leaks no information on its actual size). Each internal node contains a mercurial commitment to (a hash value of) its two children whereas each leaf node is a mercurial commitment to a database entry. The crucial idea is that internal childless nodes contain soft commitments, which keeps the commitment generation phase efficient (*i.e.*, polynomial in λ). A proof of membership for the entry x consists of a sequence of hard openings for commitments appearing in nodes on the path from leaf x to the root. Proofs of non-membership proceed similarly but rather use soft openings along the path.

As noted in [9], the above approach often results in long proofs, which may be problematic in applications, like mobile Internet connections, where users are charged depending on the number of blocks that they send/receive. To address this issue, Catalano, Fiore and Messina (CFM) suggested to increase the branching factor q of the tree and to use a primitive called *trapdoor q-mercurial commitment* (qTMC). The latter is like an ordinary mercurial commitment with the difference that it allows committing to a vector of q messages at once. With regular mercurial commitments, increasing the arity of the tree is not appropriate as generating proofs entails to reveal q values (instead of 2) at each level of the tree. However, it becomes interesting with qTMC schemes that can be opened with respect to specific vector positions without having to disclose each one of the q committed messages. The CFM construction makes use of an elegant qTMC scheme where soft commitment openings consist of a single group element, which yields dramatically shorter proofs of non-membership. On the other hand, hard openings unfortunately comprise $O(q)$ elements in the qTMC scheme described in [9]. For this reason, proofs of membership remain significantly longer than proofs of non-membership.

In this paper, we solve a problem left open in [9] and consider a primitive called *concise* mercurial vector commitment, which is a qTMC scheme allowing to commit to a q-vector in such a way that (1) hard and soft position-wise openings *both* have constant (*i.e.*, independent of q) size; (2) the committer can hard-open the commitment at position $i \in \{1, \ldots, q\}$ without revealing anything on messages at other positions in the vector. We describe a simple and natural example of such scheme. Like the CFM q-mercurial commitment, our realization relies on a specific number theoretic assumption in bilinear groups. Implementing the CFM flat-tree system with our scheme immediately yields very short proofs of membership and while retaining short proofs of non-membership. Assuming that 2^λ is a theoretical bound on the database size, we obtain proofs comprising $O(\lambda/\log(q))$ group elements for membership and non-membership. In the CFM system, proofs of membership grow as $O(\lambda \cdot q/\log(q))$, which prevents one from compressing proofs of non-membership without incurring a blow-up in the length of proofs of membership. Using our commitment scheme, both kinds of proof can be shortened by increasing q as long as the common reference string (which has size $O(q)$ as in [9]) is not too large. With $q = 128$ for instance, proofs do not exceed 2 kB in instantiations using suitable parameters.

In addition, we also show that our qTMC scheme easily lends itself to the construction of independent zero-knowledge databases. To construct such protocols satisfying a strong definition of independence, Gennaro and Micali [15] used *multi-trapdoor* mercurial commitments that can be seen as families of mercurial commitments (in the same way as multi-trapdoor commitments [14] are families of trapdoor commitments). Modulo appropriate slight modifications, our scheme can be turned into a concise multi-trapdoor qTMC scheme. It thus gives rise to the first ZK-EDB realization that simultaneously provides independence and short proofs.

ORGANIZATION. Section 2 recalls the definitions of qTMC schemes and zero-knowledge databases. We describe the new q-mercurial commitment scheme and discuss its efficiency impact in sections 3 and 4. Section 5 finally explains how the resulting ZK-EDB scheme can be made independent.

2 Background

2.1 Complexity Assumptions

We use groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p with an efficiently computable map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ such that $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$, $a, b \in \mathbb{Z}$ and $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$. In this mathematical setting, we rely on a computational assumption previously used in [5, 6].

Definition 1 ([5]). *Let $\mathbb{G}$ be a group of prime order p and $g \in \mathbb{G}$. The* **q-Diffie-Hellman Exponent** *(q-DHE) problem is, given a tuple of elements $(g, g_1, \ldots, g_q, g_{q+2}, \ldots, g_{2q})$ such that $g_i = g^{(\alpha^i)}$, for $i = 1, \ldots, q, q+2, \ldots, 2q$ and where $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, to compute the missing group element $g_{q+1} = g^{(\alpha^{q+1})}$.*

As noted in [6], this problem is not easier than the one used in [5], which is to compute $e(g, h)^{(\alpha^{q+1})}$ on input of the same values and the additional element $h \in \mathbb{G}$. The generic hardness of q-DHE is thus implied by the generic security of the family of assumptions described in [4].

2.2 Trapdoor q-Mercurial Commitments

A trapdoor q-mercurial commitment (qTMC) consists of a set of efficient algorithms (qKeygen, qHCom, qHOpen, qHVer, qSCom, qSOpen, qSVer, qFake, qHEquiv, qSEquiv) with the following specifications.

- qKeygen(λ, q): takes as input a security parameter λ and the number q of messages that can be committed to in a single commitment. The output is a pair of public/private keys (pk, tk).
- qHCom$_{pk}(m_1, \ldots, m_q)$: takes as input an ordered tuple of messages. It outputs a hard commitment C to $(m_1, \ldots, m_q)$ under the public key pk and some auxiliary state information aux.

$\mathsf{qHOpen}_{pk}(m, i, \mathsf{aux})$: is a hard opening algorithm. Given a pair $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \ldots, m_q)$, it outputs a hard de-commitment π of C w.r.t. position i if $m = m_i$. If $m \neq m_i$, it returns $\perp$.

$\mathsf{qHVer}_{pk}(m, i, C, \pi)$: is the hard verification algorithm. It outputs 1 if π gives evidence that C is a commitment to a sequence $(m_1, \ldots, m_q)$ such that $m_i = m$. Otherwise, it outputs 0.

$\mathsf{qSCom}_{pk}()$: is a probabilistic algorithm that generates a soft commitment and some auxiliary information aux. Such a commitment is not associated with a specific sequence of messages.

$\mathsf{qSOpen}_{pk}(m, i, \mathsf{flag}, \mathsf{aux})$: generates a soft de-commitment (a.k.a. "tease") τ of C to the message m at position i. The variable $\mathsf{flag} \in \{\mathbb{H}, \mathbb{S}\}$ indicates whether the state information aux corresponds to a hard commitment $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \ldots, m_q)$ or a soft one $(C, \mathsf{aux}) = \mathsf{qSCom}_{pk}()$. If $\mathsf{flag} = \mathbb{H}$ and $m \neq m_i$, the algorithm returns the error message $\perp$.

$\mathsf{qSVer}_{pk}(m, i, C, \tau)$: returns 1 if τ is a valid soft de-commitment of C to m at position i and 0 otherwise. If τ is valid and C is a hard commitment, its hard opening must be to m at index i.

$\mathsf{qFake}_{pk,tk}()$: is a randomized algorithm that takes as input the trapdoor tk and generates a q-fake commitment C and some auxiliary information aux. The commitment C is not bound to any sequence of messages. The q-fake commitment C is similar to a soft de-commitment with the difference that it can be hard-opened using the trapdoor tk.

$\mathsf{qHEquiv}_{pk,tk}(m_1, \ldots, m_q, i, \mathsf{aux})$: is a non-adaptive hard equivocation algorithm. Namely, given $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$, it generates a hard de-commitment π for C at the i^{th} position of the sequence $(m_1, \ldots, m_q)$. The algorithm is non-adaptive in that the sequence of messages has to be determined once-and-for-all before the execution of $\mathsf{qHEquiv}$.

$\mathsf{qSEquiv}_{pk,tk}(m, i, \mathsf{aux})$: is a soft equivocation algorithm. Given the auxiliary information aux returned by $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$, it creates a soft de-commitment τ to m at position i.

Standard trapdoor mercurial commitments can be seen as a special case of qTMC schemes where $q = 1$.

Correctness. The correctness requirements are similar to those of standard mercurial commitments. For any sequence $(m_1, \ldots, m_q)$, these statements must hold with overwhelming probability.

- Given a hard commitment $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \ldots, m_q)$, for all indices $i \in \{1, \ldots, q\}$, it must hold that $\mathsf{qHVer}_{pk}(m_i, i, C, \mathsf{qHOpen}_{pk}(m_i, i, \mathsf{aux})) = 1$ and $\mathsf{qSVer}_{pk}(m_i, i, C, \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathsf{aux})) = 1$.
- If $(C, \mathsf{aux}) = \mathsf{qSCom}_{pk}()$, then $\mathsf{qSVer}_{pk}(m_i, i, C, \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{S}, \mathsf{aux})) = 1$ for $i = 1, \ldots, q$.
- Given a fake commitment $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$, for each $i \in \{1, \ldots, q\}$, we must have $\mathsf{qHVer}_{pk}(m_i, i, C, \mathsf{qHEquiv}_{pk,tk}(m_1, \ldots, m_q, i, \mathsf{aux})) = 1$ and $\mathsf{qSVer}_{pk}(m_i, i, C, \mathsf{qSEquiv}_{pk,tk}(m_i, i, \mathsf{aux})) = 1$.

SECURITY. The security properties of a trapdoor q-mercurial commitment are stated as follows:

- **q-Mercurial binding:** given the public key pk, it should be computationally infeasible to output a commitment C, an index $i \in \{1, \ldots, q\}$ and pairs (m, π), (m', π') that satisfy either of these two conditions which are respectively termed "hard collision" and "soft collision":
  - $\mathsf{qHVer}_{pk}(m, i, C, \pi) = 1$, $\mathsf{qHVer}_{pk}(m', i, C, \pi') = 1$ and $m \neq m'$.
  - $\mathsf{qHVer}_{pk}(m, i, C, \pi) = 1$, $\mathsf{qSVer}_{pk}(m', i, C, \pi') = 1$ and $m \neq m'$.
- **q-Mercurial hiding:** on input of pk, no PPT adversary can find a tuple $(m_1, \ldots, m_q)$ and an index $i \in \{1, \ldots, q\}$ for which it is able to 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 $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \ldots, m_q)$, $(C', \mathsf{aux}') = \mathsf{qSCom}_{pk}()$.
- **Equivocations:** given the public key pk *and the trapdoor* tk, no PPT adversary $\mathcal{A}$ should be able to win the following games with non-negligible probability. In these games, $\mathcal{A}$ aims to distinguish the "real" world from the corresponding "ideal" one. The kind of world that $\mathcal{A}$ is faced with depends on a random $b \stackrel{R}{\leftarrow} \{0, 1\}$ flipped by the challenger. If $b = 0$, the challenger plays the "real" game and provides $\mathcal{A}$ with a real commitment/de-commitment tuple. If $b = 1$, the adversary $\mathcal{A}$ rather receives a fake commitment and equivocations. More precisely, $\mathcal{A}$ is required to guess the bit $b \in \{0, 1\}$ with no better advantage than $1/2$ in the following games:
  - **q-HHEquivocation:** when $\mathcal{A}$ chooses a message sequence $(m_1, \ldots, m_q)$, the challenger computes $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \ldots, m_q)$ if $b = 0$ and $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$ if $b = 1$. In either case, $\mathcal{A}$ receives C. When $\mathcal{A}$ chooses $i \in \{1, \ldots, q\}$, the challenger returns $\pi = \mathsf{qHOpen}_{pk}(m_i, i, \mathsf{aux})$ if $b = 0$ and $\pi = \mathsf{qHEquiv}_{pk,tk}(m_1, \ldots, m_q, i, \mathsf{aux})$ if $b = 1$.
  - **q-HSEquivocation:** when $\mathcal{A}$ chooses a message sequence $(m_1, \ldots, m_q)$, the challenger computes $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \ldots, m_q)$ if $b = 0$ and $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$ if $b = 1$. In either case, C is given to $\mathcal{A}$ who then chooses $i \in \{1, \ldots, q\}$. If $b = 0$, the challenger replies with $\tau = \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathsf{aux})$. If $b = 1$, $\mathcal{A}$ receives $\tau = \mathsf{qSEquiv}_{pk,tk}(m_i, i, \mathsf{aux})$.
  - **q-SSEquivocation:** if $b = 0$, the challenger creates a soft commitment $(C, \mathsf{aux}) = \mathsf{qSCom}_{pk}()$ and hands C to $\mathcal{A}$. If $b = 1$, $\mathcal{A}$ rather obtains a fake commitment C, which is obtained as $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk}()$. Then, $\mathcal{A}$ chooses $m \in \mathcal{M}$ and $i \in \{1, \ldots, q\}$ and gets $\tau = \mathsf{qSOpen}_{pk}(m, i, \mathbb{S}, \mathsf{aux})$ if $b = 0$ and $\tau = \mathsf{qSEquiv}_{pk,tk}(m, i, \mathsf{aux})$ if $b = 1$.

As pointed out in [8] in the case of ordinary trapdoor mercurial commitments, any qTMC scheme satisfying the q-HSEquivocation and q-SSEquivocation properties also satisfies the q-mercurial hiding requirement.

In the following, we say that a qTMC scheme is a *concise* mercurial vector commitment if the output sizes of qHOpen and qSOpen do not depend on q and if, when invoked on the index $i \in \{1, \ldots, q\}$, qHOpen does not reveal any information on messages m_j with $j \neq i$.

2.3 Zero-Knowledge Sets and Databases

An elementary database D (EDB) is a set of pairs $(x, y) \subset \{0,1\}^* \times \{0,1\}^*$, where x is called *key* and y is termed *value*. The support $[D]$ of D is the set of $x \in \{0,1\}^*$ for which there exists $y \in \{0,1\}^*$ such that $(x, y) \in D$. When $x \notin [D]$, one usually writes $D(x) = \perp$. When $x \in [D]$, the associated value $y = D(x)$ must be unique: if $(x, y) \in D$ and $(x, y') \in D$, then $y = y'$. A zero-knowledge EDB allows a prover to commit to such a database D while being able to non-interactively prove statements of the form "$x \in [D]$ and $y = D(x)$ is the associated value" or "$x \notin [D]$" without revealing any further information on D (not even the cardinality of $[D]$). Zero-knowledge sets are specific ZK-EDBs where each key is assigned the value 1.

The prover and the verifier both take as input a string σ that can be a random string (in which case, the protocol stands in the common random string model) or have a specific structure (in which case we are in the trusted parameters model). An EDB scheme is formally defined by a tuple $(\mathsf{CRS\text{-}Gen}, \mathsf{P1}, \mathsf{P2}, \mathsf{V})$ such that:

- $\mathsf{CRS\text{-}Gen}$ generates a common reference string σ on input of a security parameter λ.
- $\mathsf{P1}$ is the commitment algorithm that takes as input the database D and σ. It outputs commitment and de-commitment strings (Com, Dec).
- $\mathsf{P2}$ is the proving algorithm that, given σ, the commitment/de-commitment pair (Com, Dec) and a key $x \in \{0,1\}^*$, outputs a proof π_x.
- V is the verification algorithm that, on input of σ, Com, x and π_x, outputs either y (which must be $\perp$ if $x \notin [D]$) if it is convinced that $D(x) = y$ or *bad* if it believes that the prover is cheating.

The security requirements are formally defined in appendix A. In a nutshell, they are as follows. Correctness mandates that honestly generated proofs always satisfy the verification test. Soundness requires that provers be unable to come up with a key x and convincing proofs π_x, π'_x such that $y = \mathsf{V}(\sigma, Com, x, \pi_x) \neq \mathsf{V}(\sigma, Com, x, \pi_x) = y'$. Finally, zero-knowledge means that each proof π_x only reveals the value $D(x)$ and nothing else: for any computable database D, there must exist a simulator that outputs a simulated reference string σ' and a simulated commitment Com' that does not depend on D. For any key $x \in \{0,1\}^*$ and with oracle access to D, the simulator should be able to simulate proofs π_x that are indistinguishable from real proofs.

3 A Construction of Concise qTMC Scheme

Our idea is to build on the accumulator of Camenisch, Kohlweiss and Soriente [6], which is itself inspired by the Boneh-Gentry-Waters broadcast encryption system [5]. In the former, the public key comprises a sequence of group elements $(g, g_1, \ldots, g_q, g_{q+2}, \ldots, g_{2q})$, where q is the maximal number of accumulated values and $g_i = g^{(\alpha^i)}$ for each i. Elements of $\mathcal{V} \subseteq \{1, \ldots, q\}$ are accumulated by computing $V = \prod_{j \in \mathcal{V}} g_{q+1-j}$ and the witness for the accumulation of $i \in \mathcal{V}$ consists of $W_i = \prod_{j \in \mathcal{V} \setminus \{i\}} g_{q+1-j+i}$, which always satisfies $e(g_i, V) = e(g, W_i) \cdot e(g_1, g_q)$.

To obtain a commitment scheme, we modify this construction in order to accumulate messages $m_i \in \mathbb{Z}_p^*$ in a position-sensitive manner and we also add some randomness $\gamma \in \mathbb{Z}_p$ to have a hiding commitment. More precisely, we commit to $(m_1, \ldots, m_q)$ by computing $V = g^\gamma \cdot \prod_{j=1}^q g_{q+1-j}^{m_j}$ and obtain a kind of generalized Pedersen commitment [23]. Thanks to the specific choice of base elements however, $W_i = g_i^\gamma \cdot \prod_{j=1, j \neq i}^q g_{q+1-j+i}^{m_j}$ can serve as evidence that m_i was the i^{th} committed message as it satisfies the relation $e(g_i, V) = e(g, W_i) \cdot e(g_1, g_q)^{m_i}$. Moreover, the opening W_i at position i does not reveal anything about other components of the committed vector, which is a property that can be useful in other applications. This commitment can be proved binding under the q-DHE assumption, which would be broken if the adversary was able to produce two distinct openings of V at position i. It is also a trapdoor commitment since anyone holding $g_{q+1} = g^{(\alpha^{q+1})}$ can trapdoor open a commitment as he likes.

The scheme can be made mercurial by observing that its binding property disappears if the verification equation becomes $e(g_i, V) = e(g_1, W_i) \cdot e(g_1, g_q)^{m_i}$. The key idea is then to use commitments of the form (C, V) where $C = g^\theta$, for some $\theta \in \mathbb{Z}_p$, in hard commitments and $C = g_1^\theta$ in soft commitments. The verification equation thus becomes $e(g_i, V) = e(C, W_i) \cdot e(g_1, g_q)^{m_i}$.

Description. We assume that committed messages are elements of $\mathbb{Z}_p^*$. In practice, arbitrary messages can be committed to by first applying a collision-resistant hash function with range $\mathbb{Z}_p^*$.

qKeygen(λ, q)**:** chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ and $g \stackrel{R}{\leftarrow} \mathbb{G}$. It picks $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computes $g_1, \ldots, g_q, g_{q+2}, \ldots, g_{2q}$, where $g_i = g^{(\alpha^i)}$ for $i = 1, \ldots, q, q+2, \ldots, 2q$. The public key is defined to be $pk = \{g, g_1, \ldots, g_q, g_{q+2}, \ldots, g_{2q}\}$ and the trapdoor is $tk = g_{q+1} = g^{(\alpha^{q+1})}$.

qHCom$_{pk}(m_1, \ldots, m_q)$**:** to hard-commit to a sequence $(m_1, \ldots, m_q) \in (\mathbb{Z}_p^*)^q$, this algorithm chooses $\gamma, \theta \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and computes the commitment as the pair

$$C = g^\theta \qquad\qquad V = g^\gamma \cdot \prod_{j=1}^q g_{q+1-j}^{m_j} = g^\gamma \cdot g_q^{m_1} \cdots g_1^{m_q}.$$

The output is (C, V) and the auxiliary information is $\mathsf{aux} = (m_1, \ldots, m_q, \gamma, \theta)$.

qHOpen$_{pk}(m_i, i, \mathsf{aux})$**:** parses aux as $(m_1, \ldots, m_q, \gamma, \theta)$ and calculates

$$W_i = \Big(g_i^\gamma \cdot \prod_{j=1, j \neq i}^q g_{q+1-j+i}^{m_j}\Big)^{1/\theta}. \tag{1}$$

The hard opening of (C, V) consists of $\pi = (\theta, W_i) \in \mathbb{Z}_p \times \mathbb{G}$.

qHVer$_{pk}(m_i, i, (C, V), \pi)$**:** parses π as $(\theta, W_i) \in \mathbb{Z}_p \times \mathbb{G}$ and returns 1 if $C, V \in \mathbb{G}$ and it holds that

$$e(g_i, V) = e(C, W_i) \cdot e(g_1, g_q)^{m_i} \quad \text{and} \quad C = g^\theta. \tag{2}$$

Otherwise, it returns 0.

qSCom$_{pk}$()**:** chooses $\theta, \gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and computes $C = g_1^\theta$, $V = g_1^\gamma$. The output is (C, V) and the auxiliary information is $\mathsf{aux} = (\theta, \gamma)$.

qSOpen$_{pk}(m, i, \mathsf{flag}, \mathsf{aux})$**:** if $\mathsf{flag} = \mathbb{H}$, aux is parsed as $(m_1, \ldots, m_q, \gamma, \theta)$. The algorithm returns $\perp$ if $m \neq m_i$. Otherwise, it computes the soft opening as $W_i = \left(g_i^\gamma \cdot \prod_{j=1,j\neq i}^q g_{q+1-j+i}^{m_j}\right)^{1/\theta}$. If $\mathsf{flag} = \mathbb{S}$, the algorithm parses aux as (θ, γ) and soft-de-commits to m using $W_i = \left(g_i^\gamma \cdot g_q^{-m}\right)^{1/\theta}$. In either case, the algorithm returns $\tau = W_i \in \mathbb{G}$.

qSVer$_{pk}(m, i, (C, V), \tau)$**:** parses τ as $W_i \in \mathbb{G}$ and returns 1 if and only if it holds that $C, V \in \mathbb{G}$ and the first verification equation of (2) is satisfied.

qFake$_{pk,tk}$()**:** the fake commitment algorithm chooses $\theta, \gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and returns $(C, V) = (g^\theta, g^\gamma)$. The auxiliary information is $\mathsf{aux} = (\theta, \gamma)$.

qHEquiv$_{pk,tk}(m_1, \ldots, m_q, i, \mathsf{aux})$**:** parses aux as $(\theta, \gamma) \in (\mathbb{Z}_p)^2$. Using the trapdoor $tk = g_{q+1} \in \mathbb{G}$, it computes $W_i = \left(g_i^\gamma \cdot g_{q+1}^{-m_i}\right)^{1/\theta}$. The de-commitment consists of $\pi = (\theta, W_i)$.

qSEquiv$_{pk,tk}(m, i, \mathsf{aux})$**:** parse aux as (θ, γ) and returns $W_i = \left(g_i^\gamma \cdot g_{q+1}^{-m}\right)^{1/\theta}$.

Correctness. In hard commitments, we can check that properly generated hard de-commitments always satisfy the verification test (2) since

$$\begin{aligned}
\frac{e(g_i, V)}{e(C, W_i)} &= e\big(g^{(\alpha^i)}, g^{\gamma + \sum_{j=1}^q m_j (\alpha^{q+1-j})}\big) / e\big(g^\theta, g^{(\gamma(\alpha^i) + \sum_{j=1, j\neq i}^q m_j (\alpha^{q+1-j+i}))/\theta}\big) \\
&= e\big(g, g^{\gamma(\alpha^i) + \sum_{j=1}^q m_j (\alpha^{q+1-j+i})}\big) / e\big(g, g^{\gamma(\alpha^i) + \sum_{j=1, j\neq i}^q m_j (\alpha^{q+1-j+i})}\big) \\
&= e(g, g)^{m_i (\alpha^{q+1})} = e(g_1, g_q)^{m_i}.
\end{aligned}$$

As for soft commitments, soft de-commitments always satisfy the first relation of (2) since

$$\begin{aligned}
e(C, W_i) \cdot e(g_1, g_q)^{m_i} &= e\big(g_1^\theta, (g_i^\gamma \cdot g_q^{-m_i})^{1/\theta}\big) \cdot e(g_1, g_q)^{m_i} \\
&= e\big(g_1, g_i^\gamma \cdot g_q^{-m_i}\big) \cdot e(g_1, g_q)^{m_i} = e(g_1^\gamma, g_i) = e(g_i, V).
\end{aligned}$$

We finally observe that, in any fake commitment $(C, V) = (g^\theta, g^\gamma)$, the hard de-commitment (θ, W_i) successfully passes the verification test as

$$\begin{aligned}
e(C, W_i) \cdot e(g_1, g_q)^{m_i} &= e\big(g^\theta, (g_i^\gamma \cdot g_{q+1}^{-m_i})^{1/\theta}\big) \cdot e(g_1, g_q)^{m_i} \\
&= e\big(g, g_i^\gamma \cdot g_{q+1}^{-m_i}\big) \cdot e(g_1, g_q)^{m_i} = e(g_i, g^\gamma) = e(g_i, V).
\end{aligned}$$

Security. To prove the security of the scheme, we first notice that it is a "proper" qTMC [8] since, in hard commitments, the soft de-commitment is a proper subset of the hard de-commitment.

Theorem 1. *The above scheme is a secure concise qTMC if the q-DHE assumption holds in* $\mathbb{G}$*.*

Proof. We first show the q-mercurial binding property. Let us assume that, given the public key, an adversary $\mathcal{A}$ is able to generate soft collisions (since the scheme is "proper", the case of hard collisions immediately follows). That is, $\mathcal{A}$ comes up with a commitment $(C, V) \in \mathbb{G}^2$, an index $i \in \{1, \ldots, q\}$, a valid hard decommitment $\pi = (\theta, W_i) \in \mathbb{Z}_p \times \mathbb{G}$ to m_i at position i and a valid soft decommitment $\tau = W_i' \in \mathbb{G}$ to m_i' such that $m_i \neq m_i'$. We must have

$$e(g_i, V) = e(g^\theta, W_i) \cdot e(g_1, g_q)^{m_i} \qquad e(g_i, V) = e(g^\theta, W_i') \cdot e(g_1, g_q)^{m_i'},$$

so that $e\big(g^\theta, W_i/W_i'\big) = e(g_1, q_q)^{m_i' - m_i}$ and $e\big(g, (W_i/W_i')^{\theta/(m_i'-m_i)}\big) = e(g_1, g_q)$. Since $m_i \neq m_i'$, the latter relation implies that $g_{q+1} = (W_i/W_i')^{\theta/(m_i'-m_i)}$ is revealed by the soft collision, which contradicts the q-DHE assumption.

We now turn to the q-HHE, q-HSE and q-SSE equivocation properties (which imply q-mercurial hiding). A fake commitment has the form $(C, V) = (g^\theta, g^\gamma)$ and its hard equivocation to (m_i, i) is the pair $\big(\theta, W_i = (g_i^\gamma \cdot g_{q+1}^{-m_i})^{1/\theta}\big)$. For any sequence of messages $(m_1, \ldots, m_q) \in (\mathbb{Z}_p^*)^q$, there exists $\gamma' \in \mathbb{Z}_p$ such that

$$V = g^{\gamma'} \cdot \prod_{j=1}^{q} g_{q+1-j}^{m_j}. \tag{3}$$

Then, the corresponding hard opening of (C, V) w.r.t. m_i at position i should be obtained as $W_i' = (g_i^{\gamma'} \cdot \prod_{j=1, j\neq i}^{q} g_{q+1-j+i}^{m_j})^{1/\theta}$. Since V also equals g^γ, if we raise both members of (3) to the power α^i, we find that

$$g_i^\gamma = g_i^{\gamma'} \cdot \prod_{j=1}^{q} g_{q+1-j+i}^{m_j}.$$

Therefore, the element $W_i = (g_i^\gamma \cdot g_{q+1}^{-m_i})^{1/\theta}$ returned by the hard equivocation algorithm can also be written $W_i = (g_i^{\gamma'} \cdot \prod_{j=1, j\neq i}^{q} g_{q+1-j+i}^{m_j})^{1/\theta}$. It comes that fake commitments and hard equivocations have exactly the same distribution as hard commitments and their hard openings.

The q-HSEquivocation property follows from the above arguments (since the scheme is "proper"). To prove the indistinguishability in the q-SSEquivocation game, we note that fake commitments $(C, V) = (g^\theta, g^\gamma)$ have the same distribution as soft ones as they can be written $(C, V) = (g_1^{\tilde{\theta}}, g_1^{\tilde{\gamma}})$ where $\tilde{\theta} = \theta/\alpha$ and $\tilde{\gamma} = \gamma/\alpha$. Their soft equivocation $W_i = (g_i^\gamma \cdot g_{q+1}^{-m_i})^{1/\theta}$ can be written $(g_i^{\alpha\tilde{\gamma}} \cdot g_{q+1}^{-m_i})^{1/(\alpha\tilde{\theta})} = (g_i^{\tilde{\gamma}} \cdot g_q^{-m_i})^{1/\tilde{\theta}}$ and has the distribution of a soft opening. □

Instantiation with Asymmetric Pairings. It is simple[3] to describe the construction in terms of asymmetric pairings $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$, where $\mathbb{G} \neq \hat{\mathbb{G}}$ and an isomorphism $\psi : \hat{\mathbb{G}} \rightarrow \mathbb{G}$ is efficiently computable. The public key comprises generators $\hat{g} \in \hat{\mathbb{G}}$ and $\hat{g}_i$ for $i = 1, \ldots, q, q+2, \ldots, 2q$. Then, hard (resp. soft) commitments $(C, V) \in \hat{\mathbb{G}} \times \mathbb{G}$ are pairs of group elements obtained as $C = \hat{g}^\theta$ and

[3] The security then relies on the hardness of computing $\psi(\hat{g})^{(\alpha^{q+1})}$ on input of $(\hat{g}, \hat{g}_1, \ldots, \hat{g}_q, \hat{g}_{q+2}, \ldots, \hat{g}_{2q}) \in \hat{\mathbb{G}}^{2q}$, where $\hat{g}_i = \hat{g}^{(\alpha^i)}$ for each i.

$V = \psi(\hat{g})^{\gamma} \cdot \prod_{j=1}^{q} \psi(\hat{g}_{q+1-j})^{m_j}$ (resp. $C = \hat{g}_1^{\theta}$ and $V = \psi(\hat{g}_1)^{\gamma}$). Hard openings are pairs $(\theta, W_i) \in \mathbb{Z}_p^* \times \mathbb{G}$, where $W_i = \psi(\hat{g}_i)^{\gamma/\theta} \cdot \prod_{j=1, j \neq i}^{q} \psi(\hat{g}_{q+1-j+i})^{m_j/\theta}$ and they are verified by checking that $C = \hat{g}^{\theta}$ and $e(V, \hat{g}_i) = e(W_i, C) \cdot e(\psi(\hat{g}_1), \hat{g}_q)^{m_i}$. Using the trapdoor $\hat{g}_{q+1}$, fake commitments $(C, V) = (\hat{g}^{\theta}, \psi(\hat{g})^{\gamma})$ can be equivocated by outputting θ and $W_i = \psi(\hat{g}_i)^{\gamma/\theta} \cdot \psi(\hat{g}_{q+1})^{-m_i/\theta}$.

4 Implications on the Efficiency of ZK-EDBs

The construction [9] of ZK-EDB from qTMC schemes goes as follows. Each key x is assigned to a leaf of a q-ary tree of height h (and can be seen as the label of the leaf, expressed in q-ary encoding), so that q^h is the theoretical bound on the size of the EDB.

The committing phase is made efficient by pruning subtrees where all leaves correspond to keys that are *not* in the database. Only the roots (called "frontier nodes" and at least one sibling of which is an ancestor of a leaf in the EDB) of these subtrees are kept in the tree and contain soft q-commitments. For each key x such that $D(x) \neq \perp$, the corresponding leaf contains a standard hard mercurial commitment to a hash value of $D(x)$. As for remaining nodes, each internal one contains a hard q-commitment to messages obtained by hashing its children. The q-commitment at the root then serves as a commitment to the entire EDB.

To convince a verifier that $D(x) = v \neq \perp$ for some key x, the prover generates a proof of membership consisting of hard openings for commitments in nodes on the path connecting leaf x to the root. At each level of the tree, the q-commitment is hard-opened with respect to the position determined by the q-ary encoding of x at that level.

To provide evidence that some key x does not belong to the database (in other words, $D(x) = \perp$), the prover first generates the missing portion of the subtree where x lies. Then, it reveals soft openings for all (hard or soft) commitments contained in nodes appearing in the path from x to the root.

As in the original zero-knowledge EDB construction [21], only storing commitments in subtrees containing leaves x for which $D(x) \neq \perp$ (and soft commitments at nodes that have no descendants) is what allows committing with complexity $O(h \cdot |D|)$ instead of $O(q^h)$.

The advantage of using qTMC schemes and q-ary (with $q > 2$) trees lies in that proofs can be made much shorter if, at each level, commitments can be opened w.r.t. the required position $i \in \{1, \ldots, q\}$ without having to reveal q values. The qTMC scheme of [9] features soft openings consisting of a single group element and, for an appropriate branching factor q, allows reducing proofs of non-membership by 73% in comparison with [21]. On the other hand, hard openings still have length $O(q)$ and proofs of membership thus remain significantly longer than proofs of non-membership. If h denotes the height of the tree, the former consist of $h(q + 4) + 5$ elements of $\mathbb{G}$ (in an implementation with asymmetric pairings) while the latter only demand $4h + 4$ such elements.

If we plug our qTMC scheme into the above construction, proofs of membership become essentially as short as proofs of non-membership. At each internal node, each hard opening only requires to reveal $(C, V) \in \hat{\mathbb{G}} \times \mathbb{G}$ and $(\theta, W_i) \in \mathbb{Z}_p \times \mathbb{G}$. At the same time, proofs of non-membership remain as short as in [9] since, at each internal node, the prover only discloses (C, V) and W_i.

To concretely assess proof sizes, we assume (as in [9]) that elements of $\hat{\mathbb{G}}$

q	h	Membership	Non-Membership	Membership in [9]
8	43	220	176	521
16	32	165	132	643
32	26	135	108	941
64	22	115	92	1501
128	19	100	80	2513

Fig. 1. Required number of group elements per proof

count as two elements of $\mathbb{G}$ (since their representation is usually twice as large using suitable parameters and optimizations such as those of [2]), each one of which costs $|p|$ bits to represent. Then, we find that proofs of membership and non-membership eventually amount to $5h + 5$ and $4h + 4$ elements of $\mathbb{G}$, respectively. These short hard openings allow us to increase the branching factor of the tree as long as the length of the common reference string is deemed acceptable.

The table of figure 1 summarizes the proof lengths (expressed in numbers of $\mathbb{G}$ elements and in comparison with [9]) for various branching factors and assuming that $q^h \approx 2^{128}$ theoretically bounds the EDB's size. In the MRK construction, membership (resp. non-membership) can be proved using 773 (resp. 644) group elements. The best tradeoff achieved in [9] was for $q = 8$, where proofs of non-membership could be reduced to 176 elements but proofs of membership still took 521 elements. With $q = 8$, we have equally short proofs of non-membership and only need 220 elements to prove membership, which improves CFM [9] by about 57% and MRK [21] by 71%.

Moreover, we can shorten both kinds of proof by increasing q: with $q = 128$ for instance, no more than 100 group elements (or 13% of the original length achieved in [21]) are needed to prove membership whereas 2513 elements are necessary in [9]. Instantiating our scheme with curves of [2] yields proofs of less than 2 kB when $q = 128$. For such relatively small values of q, Cheon's attack [12] does not require to increase the security parameter λ and it is reasonable to use groups $(\mathbb{G}, \hat{\mathbb{G}})$ where elements of $\mathbb{G}$ have a 161-bit representation.

5 Achieving Strong Independence

In [15], Gennaro and Micali formalized the notion of *independent zero-knowledge* EDBs which requires that adversaries be unable to correlate their database to those created by honest provers.

The strongest flavor of independence considers two-stage adversaries $\mathcal{A} =$

$(\mathcal{A}_1, \mathcal{A}_2)$. First, $\mathcal{A}_1$ observes ℓ honest provers' commitments $(Com_1, \ldots, Com_\ell)$ and queries proofs for keys of her choice in underlying databases $D_1, \ldots, D_\ell$ before outputting her own commitment Com. Then, two copies of $\mathcal{A}_2$ are executed: in the first one, $\mathcal{A}_2$ is given oracle access to provers that "open" Com_i w.r.t D_i whereas, in the second run, $\mathcal{A}_2$ has access to provers for different[4] databases D'_i that agree with D_i for the set Q_i of queries made by $\mathcal{A}_1$. Eventually, both executions of $\mathcal{A}_2$ end with $\mathcal{A}_2$ outputting a key x, which is identical in both runs, and a proof π_x. The resulting database value $D(x)$ is required to be the same in the two copies, meaning that it was fixed at the end of the committing stage.

In the strongest definition of [15], $\mathcal{A}_1$ is allowed to copy one of the honest provers' commitment (say Com_i) as long as the key x returned by $\mathcal{A}_2$ is never queried to $\mathsf{Sim}_2(St_i, Com_i)$ by $\mathcal{A}_1$ or $\mathcal{A}_2$: in other words, $\mathcal{A}_2$'s answer must be fixed on all values x that were not queried to the i^{th} prover.

Definition 2. *[15] A ZK-EDB protocol is strongly independent if, for any polynomial ℓ, any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and any databases $D_1, \ldots, D_\ell$, $D'_1, \ldots, D'_\ell$, the following probability is negligible.*

$$\begin{aligned}
\Pr\Big[&(\sigma, St_0) \leftarrow \mathsf{Sim}_0(\lambda);\ (Com_i, St_i) \leftarrow \mathsf{Sim}_1(St_0)\ \forall i = 1, \ldots, \ell; \\
&(Com, \omega) \leftarrow \mathcal{A}_1^{\mathsf{Sim}_2^{D_i(\cdot)}(St_i, Com_i)}(\sigma, Com_1, \ldots, Com_\ell); \\
&(x, \pi_x) \leftarrow \mathcal{A}_2^{\mathsf{Sim}_2^{D_i(\cdot)}(St_i, Com_i)}(\sigma, \omega);\ (x, \pi'_x) \leftarrow \mathcal{A}_2^{\mathsf{Sim}_2^{D'_i \dashv_{Q_i} D_i(\cdot)}(St_i, Com_i)}(\sigma, \omega); \\
&\big(bad \neq \mathsf{V}(\sigma, Com, x, \pi_x) \neq \mathsf{V}(\sigma, Com, x, \pi'_x) \neq bad\big) \wedge \Big((\forall i : Com \neq Com_i) \\
&\quad \vee \big(\exists i : (Com = Com_i) \wedge (x \notin Q_i \cup Q'_i)\big)\Big)\Big],
\end{aligned}$$

where Q_i (resp. Q'_i) stands for the list of queries made by $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) to $\mathsf{Sim}_2^{D_i(\cdot)}(St_i, Com_i)$ (resp. $\mathsf{Sim}_2^{D_i(\cdot)}(St_i, Com_i)$ and $\mathsf{Sim}_2^{D'_i \dashv_{Q_i} D_i(\cdot)}(St_i, Com_i)$) and $D'_i \dashv_{Q_i} D_i$ denotes a database that agrees with D'_i on all keys but those in Q_i where it agrees with D_i.

An efficient construction of independent ZK-EDB was proved in [15] to satisfy the above definition under the strong RSA assumption. It was obtained by extending Gennaro's multi-trapdoor commitment scheme [14] and making it mercurial.

We show how to turn our qTMC scheme into a multi-trapdoor q-mercurial commitment scheme that yields strongly independent EDBs with short proofs.

MULTI-TRAPDOOR Q-MERCURIAL COMMITMENTS. A multi-trapdoor qTMC can be seen as extending qTMC schemes in the same way as multi-trapdoor commitments generalize ordinary trapdoor commitments. It can be defined as a family of trapdoor q-mercurial commitments, each member of which is identified by a string tag and has its own trapdoor tk_{tag}. The latter is generated from tag using a master trapdoor TK that matches the master public key PK.

[4] For this reason, commitments $(Com_1, \ldots, Com_\ell)$ are produced using the ZK-EDB simulator, whose definition is recalled in appendix A, as the two executions of $\mathcal{A}_2$ proceed as if underlying databases were different.

$\mathsf{qKeygen}(\lambda, q)$: has the same specification as in section 2.2 but, in addition to the master key pair (PK, TK), it outputs the description of a tag space $\mathcal{T}$.

$\mathsf{qHCom}_{PK}(m_1, \ldots, m_q, tag)$: given an ordered tuple $(m_1, \ldots, m_q)$ and $tag \in \mathcal{T}$, this algorithm outputs a hard commitment C under (PK, tag) and some auxiliary state information aux.

$\mathsf{qHOpen}_{PK}(m, i, tag, \mathsf{aux})$: given a pair $(C, \mathsf{aux}) = \mathsf{qHCom}_{PK}(m_1, \ldots, m_q, tag)$, this algorithm outputs a hard de-commitment π of C w.r.t. position i if $m = m_i$. If $m \neq m_i$, it returns $\perp$.

$\mathsf{qHVer}_{PK}(m, i, C, tag, \pi)$: outputs 1 if and only if π gives evidence that, under the tag tag, C is bound to a sequence $(m_1, \ldots, m_q)$ such that $m_i = m$.

$\mathsf{qSCom}_{PK}()$: generates a soft commitment and some auxiliary information aux. Such a commitment is not associated with any specific messages or tag.

$\mathsf{qSOpen}_{PK}(m, i, \mathsf{flag}, tag, \mathsf{aux})$: generates a soft de-commitment τ of C to m at position i and w.r.t. tag. The variable $\mathsf{flag} \in \{\mathbb{H}, \mathbb{S}\}$ indicates whether τ pertains to a hard commitment $(C, \mathsf{aux}) = \mathsf{qHCom}_{PK}(m_1, \ldots, m_q, tag)$ or a soft commitment $(C, \mathsf{aux}) = \mathsf{qSCom}_{PK}()$. If $\mathsf{flag} = \mathbb{H}$ and $m \neq m_i$, the algorithm returns $\perp$.

$\mathsf{qSVer}_{PK}(m, i, C, \tau, tag)$ returns 1 if, under $tag \in \mathcal{T}$, τ is deemed as a valid soft de-commitment of C to m at position i and 0 otherwise.

$\mathsf{qTrapGen}_{PK,TK}(tag)$: given a string $tag \in \mathcal{T}$, this algorithm generates a tag-specific trapdoor tk_{tag} using the master trapdoor TK.

$\mathsf{qFake}_{PK,tk_{tag}}()$: outputs a q-fake commitment C and some auxiliary state information aux.

$\mathsf{qHEquiv}_{PK,tk_{tag}}(m_1, \ldots, m_q, i, tag, \mathsf{aux})$: given $(C, \mathsf{aux}) = \mathsf{qFake}_{PK,tk_{tag}}()$, this algorithm generates a hard de-commitment π for C and $tag \in \mathcal{T}$ at the i^{th} position of the sequence $(m_1, \ldots, m_q)$. The sequence of messages has to be determined once-and-for-all before the execution of $\mathsf{qHEquiv}$.

$\mathsf{qSEquiv}_{PK,tk_{tag}}(m, i, tag, \mathsf{aux})$: using the trapdoor tk_{tag} and the state information aux returned by $(C, \mathsf{aux}) = \mathsf{qFake}_{PK,tk_{tag}}()$, this algorithm creates a soft de-commitment τ to m at position i and w.r.t. $tag \in \mathcal{T}$.

Again, we call such a scheme *concise* if it satisfies the same conditions as those mentioned at the end of section 2.2.

The security properties are expressed by naturally requiring the q-mercurial hiding and equivocation properties to hold for each $tag \in \mathcal{T}$. In equivocation games, the adversary should be unable to distinguish the two games even knowing the master trapdoor TK. As for the q-mercurial binding property, it states that no PPT adversary $\mathcal{A}$ should have non-negligible advantage in this game:

q-Mercurial binding game: $\mathcal{A}$ chooses strings $tag_1, \ldots, tag_\ell \in \mathcal{T}$. Then, the challenger generates a master key pair $(TK, PK) \leftarrow \mathsf{qKeygen}(\lambda, q)$ and gives PK to $\mathcal{A}$ who starts invoking a trapdoor oracle $\mathcal{TG}$: the latter receives $tag \in \{tag_1, \ldots, tag_\ell\}$ and returns $tk_{tag} \leftarrow \mathsf{qTrapGen}_{PK,TK}(tag)$. Eventually, $\mathcal{A}$ chooses a family $tag^\star \in \mathcal{T} \backslash \{tag_1, \ldots, tag_\ell\}$ for which she aims to generate a collision: she wins if she outputs C, an index $i \in \{1, \ldots, q\}$ and pairs (m, π), (m', π') (resp. (m, π) and (m', τ)) such that $\mathsf{qHVer}_{PK}(m, i, C, tag^\star, \pi) = 1$

and $\mathsf{qHVer}_{PK}(m', i, C, tag^\star, \pi') = 1$ (resp. $\mathsf{qHVer}_{PK}(m, i, C, tag^\star, \pi) = 1$ and $\mathsf{qSVer}_{PK}(m', i, C, tag^\star, \tau) = 1$) but $m \neq m'$.

As in [14], the latter definition captures security in a non-adaptive sense in that the adversary chooses $tag_1, \ldots, tag_\ell$ before seeing the public key PK. As noted in [13, 19] in the case of ordinary multi-trapdoor commitments, some applications might require to consider a notion of adaptive security where, much in the fashion of identity-based trapdoor commitments [1, 7], the adversary can query $\mathcal{TG}$ in an adaptive fashion. In the present context, non-adaptive security suffices.

A Construction of Multi-Trapdoor qTMC. The construction combines the qTMC scheme of section 3 with a programmable hash function $H_{\mathbb{G}} : \mathcal{T} \to \mathbb{G}$ and techniques that were introduced in [3]. Programmable hash functions, as formalized by Hofheinz and Kiltz [17], are designed in such a way that a trapdoor information makes it possible to relate the output $H_{\mathbb{G}}(M)$, which lies in a group $\mathbb{G}$, to computable values $a_M, b_M \in \mathbb{Z}_p$ satisfying $H_{\mathbb{G}}(M) = g^{a_M} \cdot h^{b_M}$. Informally (see [17] for a formal definition), a (m, n)-programmable hash function is such that, for any $M_1, \ldots, M_m$, $M'_1, \ldots, M'_n$ such that $M_i \neq M'_j$, there is a non-negligible probability that $b_{M_i} = 0$ and $b_{M'_j} \neq 0$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The number theoretic hash function used in [11, 25] is an example of such a $(1, \ell)$-programmable hash function, for some polynomial ℓ.

qKeygen(λ, q)**:** is as in section 3 but the algorithm also chooses a tag space $\mathcal{T} = \{0, 1\}^L$ and a $(1, \ell)$-programmable hash function $H_{\mathbb{G}} : \mathcal{T} \to \mathbb{G}$ for some polynomials ℓ, L. The public key is $PK = \{\mathcal{T}, g, g_1, \ldots, g_q, g_{q+2}, \ldots, g_{2q}, H_{\mathbb{G}}\}$ and the master trapdoor is $TK = g_{q+1} = g^{(\alpha^{q+1})}$.

qHCom$_{PK}(m_1, \ldots, m_q, tag)$**:** to hard-commit to a sequence $(m_1, \ldots, m_q) \in (\mathbb{Z}_p^*)^q$, this algorithm chooses $\gamma, \theta \xleftarrow{R} \mathbb{Z}_p$ and computes $(C, V) = \big(g^\theta, g^\gamma \cdot \prod_{j=1}^{q} g_{q+1-j}^{m_j}\big)$. The output is (C, V) and the auxiliary information is $\mathsf{aux} = (m_1, \ldots, m_q, \gamma, \theta)$.

qHOpen$_{PK}(m_i, i, tag, \mathsf{aux})$**:** parses aux as $(m_1, \ldots, m_q, \gamma, \theta)$, chooses $r \xleftarrow{R} \mathbb{Z}_p^*$ and computes

$$(W_i, Z_i) = \Big(\big(g_i^\gamma \cdot \prod_{j=1, j\neq i}^{q} g_{q+1-j+i}^{m_j} \cdot H_{\mathbb{G}}(tag)^r\big)^{1/\theta}, g^{-r}\Big), \qquad (4)$$

The hard opening of (C, V) with respect to $tag \in \mathcal{T}$ consists of the triple $\pi = (\theta, W_i, Z_i) \in \mathbb{Z}_p \times \mathbb{G}^2$.

qHVer$_{PK}(m_i, i, (C, V), tag, \pi)$**:** parses π as $(\theta, W_i, Z_i) \in \mathbb{Z}_p \times \mathbb{G}^2$ and returns 1 if $C, V \in \mathbb{G}$ and relations (5) are both satisfied. Otherwise, it returns 0.

$$e(g_i, V) = e(C, W_i) \cdot e(g_1, g_q)^{m_i} \cdot e(H_{\mathbb{G}}(tag), Z_i) \qquad C = g^\theta. \quad (5)$$

qSCom$_{PK}()$**:** chooses $\theta, \gamma \xleftarrow{R} \mathbb{Z}_p$ and computes $C = g_1^\theta$, $V = g_1^\gamma$. The output is (C, V) and the auxiliary information is $\mathsf{aux} = (\theta, \gamma)$.

qSOpen$_{PK}(m, i, \mathsf{flag}, tag, \mathsf{aux})$**:** if $\mathsf{flag} = \mathbb{H}$, aux is parsed as $(m_1, \ldots, m_q, \gamma, \theta)$. The algorithm returns $\perp$ if $m \neq m_i$. Otherwise, the soft opening $\tau = (W_i, Z_i)$

is generated as per (4). If $\mathsf{flag} = \mathbb{S}$, the algorithm parses aux as (θ, γ) and soft-decommits to m using

$$(W_i, Z_i) = \Big(\big(g_i^{\gamma} \cdot g_q^{-m} \cdot H_{\mathbb{G}}(tag)^r \big)^{1/\theta}, g_1^{-r} \Big), \tag{6}$$

where $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$. In either case, the algorithm returns $\tau = (W_i, Z_i) \in \mathbb{G}^2$.

$\mathsf{qSVer}_{pk}(m, i, (C, V), \tau, tag)$: parses τ as $(W_i, Z_i) \in \mathbb{G}$ and returns 1 if and only if $C, V \in \mathbb{G}$ and the first verification equation of (5) is satisfied.

$\mathsf{qTrapGen}_{PK,TK}(tag)$: given $TK = g_{q+1}$, a trapdoor for $tag \in \mathcal{T}$ is computed $tk_{tag} = (t_{tag,1}, t_{tag,2}) = (g_{q+1} \cdot H_{\mathbb{G}}(tag)^s, g^{-s})$ for a random $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$.

$\mathsf{qFake}_{PK,tk_{tag}}()$: outputs a pair $(C, V) = (g^{\theta}, g^{\gamma})$, where $\theta, \gamma \overset{R}{\leftarrow} \mathbb{Z}_p^*$, and retains the state information $\mathsf{aux} = (\theta, \gamma)$.

$\mathsf{qHEquiv}_{PK,tk_{tag}}(m_1, \ldots, m_q, i, tag, \mathsf{aux})$: parses aux as $(\theta, \gamma) \in (\mathbb{Z}_p^*)^2$ and the trapdoor tk_{tag} as $(t_{tag,1}, t_{tag,2}) \in \mathbb{G}^2$. It randomly picks $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and computes $(W_i, Z_i) = \Big(\big(g_i^{\gamma} \cdot t_{tag,1}^{-m_i} \cdot H_{\mathbb{G}}(tag)^r \big)^{1/\theta}, t_{tag,2}^{-m_i} \cdot g^{-r} \Big)$. The de-commitment is $\pi = (\theta, W_i, Z_i) = \Big(\theta, \big(g_i^{\gamma} \cdot g_{q+1}^{-m_i} \cdot H_{\mathbb{G}}(tag)^{r'} \big)^{1/\theta}, g^{-r'} \Big)$, where $r' = -s m_i + r$.

$\mathsf{qSEquiv}_{PK,tk_{tag}}(m, i, tag, \mathsf{aux})$: parse aux as (θ, γ) and computes (W_i, Z_i) as in $\mathsf{qHEquiv}_{PK,tk_{tag}}$.

Theorem 2. *The scheme is a concise multi-trapdoor qTMC if the q-DHE assumption holds.*

Proof. Given in the full version of the paper. □

Strongly Independent ZK-EDBs from Multi-Trapdoor qTMC. Following [15], a multi-trapdoor qTMC can be combined with a digital signature and a collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathcal{T}$ to give a strongly independent ZK-EDB. To commit to a database D, the prover first generates a key pair $(\mathsf{SK}, \mathsf{VK})$ for an existentially unforgeable (in the sense of [16]) signature scheme $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ [16]. The commitment string is (Com, VK), where all commitments are produced using the qTMC family (with $q = 1$ at the leaves and $q > 1$ at internal nodes) indexed by the tag $H(\mathsf{VK})$. To generate a proof for some key x, the prover generates a proof π_x (by opening the appropriate commitments using Dec) and outputs π_x and $\mathsf{sig}_x = \mathcal{S}(\mathsf{SK}, (Com, x))$. Verification entails to check π_x and that $\mathcal{V}(\mathsf{sig}_x, \mathsf{VK}, (Com, x)) = 1$. The security proof of this scheme (detailed in the full version of the paper) is similar to that of theorem 3 in [15].

References

1. G. Ateniese, B. de Medeiros. Identity-Based Chameleon Hash and Applications. In *Financial Cryptography'04*, *LNCS* 3110, pp. 164–180, 2004.
2. P. Barreto, M. Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. In *SAC'05*, *LNCS* 3897, pp. 319–331, 2005.
3. D. Boneh, X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Eurocrypt'04*, *LNCS* 3027, pp. 223–238, 2004.

4. D. Boneh, X. Boyen, E.-J. Goh. Hierarchical Identity-Based encryption with Constant Size Ciphertext. In *Eurocrypt'05*, *LNCS* 3494, pp. 440–456, 2005.
5. D. Boneh, C. Gentry, B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Crypto'05*, *LNCS* 3621, pp. 258–275, 2005.
6. J. Camenisch, M. Kohlweiss, C. Soriente. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In *PKC'09*, *LNCS* 5443, pp. 481–500, 2009.
7. R. Canetti, Y. Dodis, R. Pass, S. Walfish. Universally Composable Security with Global Setup. In *TCC'07*, *LNCS* 4392, pp. 61–85, 2007.
8. D. Catalano, Y. Dodis, I. Visconti. Mercurial Commitments: Minimal Assumptions and Efficient Constructions. In *TCC'06*, *LNCS* 3876, pp. 120–144, 2006.
9. D. Catalano, D. Fiore, M. Messina. Zero-Knowledge Sets with Short Proofs. In *Eurocrypt'08*, *LNCS* 4965, pp. 433–450, 2008.
10. M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, L. Reyzin. Mercurial Commitments with Applications to Zero-Knowledge Sets. In *Eurocrypt'05*, *LNCS* 3494, pp. 422–439, 2005.
11. D. Chaum, J.-H. Evertse, J. van de Graaf. An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In *Eurocrypt'87*, *LNCS* 304, pp. 127–141, 1987.
12. J. H. Cheon. Security Analysis of the Strong Diffie-Hellman Problem. In *Eurocrypt'06*, *LNCS* 4004, pp. 1–11, 2006.
13. M. Di Raimondo, R. Gennaro. New Approaches for Deniable Authentication. In *ACM-CCS'05*, pp. 112–121, 2005.
14. R. Gennaro. Multi-trapdoor Commitments and Their Applications to Proofs of Knowledge Secure Under Concurrent Man-in-the-Middle Attacks. In *Crypto'04*, *LNCS* 3152, pp. 220–236, 2004.
15. R. Gennaro, S. Micali. Independent Zero-Knowledge Sets. In *ICALP'06*, *LNCS* 4052, pp. 34–45, 2006.
16. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen message attacks. *SIAM J. of Computing*, 17(2):281–308, 1988.
17. D. Hofheinz, E. Kiltz. Programmable Hash Functions and Their Applications. In *Crypto'08*, *LNCS* 5157, pages 21–38. Springer, 2008.
18. M. Liskov. Updatable Zero-Knowledge Databases. In *Asiacrypt'05*, *LNCS* 3788, pp. 174–198, 2005.
19. P. MacKenzie, K. Yang. On Simulation-Sound Trapdoor Commitments. In *Eurocrypt'04*, *LNCS* 3027, pp. 382–400, 2004.
20. R. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Crypto'88*, *LNCS* 403, pp. 369–378, 1988.
21. S. Micali, M.-O. Rabin, J. Kilian. Zero-Knowledge Sets. In *FOCS'03*, pp. 80–91, 2003.
22. R. Ostrovsky, C. Rackoff, A. Smith. Efficient Consistency Proofs for Generalized Queries on a Committed Database. In *ICALP'04*, *LNCS* 3142, pp. 1041–1053, 2004.
23. T. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Crypto'91*, *LNCS* 576, pp. 129–140, 1991.
24. M. Prabhakaran, R. Xue. Statistically Hiding Sets. In *CT-RSA'09*, *LNCS* 5473, pp. 100–116, 2009.
25. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Eurocrypt'05*, *LNCS* 3494, pp. 114–127, 2005.

A Security Properties of Zero-Knowledge Databases

The completeness, soundness and zero-knowledge properties of ZK-EDBs are formally stated as follows.

Completeness: For all databases D and for all keys x, it must hold that

$$\Pr\big[\sigma \leftarrow \mathsf{CRS\text{-}Gen}(\lambda); (Com, Dec) \leftarrow \mathsf{P1}(\sigma, D);\\ \pi_x \leftarrow \mathsf{P2}(\sigma, D, Com, Dec, x) : \mathsf{V}(\sigma, Com, x, \pi_x) = D(x)\ \big] = 1 - \nu.$$

for some negligible function ν.

Soundness: For all keys x and for any probabilistic poly-time algorithm P', the following probability is negligible:

$$\Pr\big[\sigma \leftarrow \mathsf{CRS\text{-}Gen}(\lambda); (Com, x, \pi_x, \pi'_x) \leftarrow \mathsf{P}'(\sigma, D);\\ \mathsf{V}(\sigma, Com, x, \pi_x) = y \neq bad \wedge \mathsf{V}(\sigma, Com, x, \pi'_x) = y' \neq bad \wedge (y \neq y')\ \big].$$

Zero-knowledge: for any PPT adversary $\mathcal{A}$ and any efficiently computable database D, there must exist an efficient simulator $(\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2^D)$ such that the outputs of the following experiments are indistinguishable:

Real experiment:

1. Set $\sigma \leftarrow \mathsf{CRS\text{-}Gen}(\lambda)$, $(Com, Dec) \leftarrow \mathsf{P1}(\sigma, D)$ and $s_0 = \varepsilon$, $\pi_0 = \varepsilon$.
2. For $i = 1, \ldots, n$, $\mathcal{A}$ outputs $(x_i, s_i) \leftarrow \mathcal{A}(\sigma, Com, \pi_0, \ldots, \pi_{i-1}, s_{i-1})$ and obtains a real proof $\pi_i = \mathsf{P2}(\sigma, D, Com, Dec, x_i)$.

The output is $(\sigma, x_1, \pi_1, \ldots, x_n, \pi_n)$.

Ideal experiment:

1. Set $(\sigma', St_0) \leftarrow \mathsf{Sim}_0(\lambda)$, $(Com', St_1) \leftarrow \mathsf{Sim}_1(St_0)$ as well as $s_0 = \varepsilon$, $\pi'_0 = \varepsilon$.
2. For $i = 1, \ldots, n$, $\mathcal{A}$ outputs $(x_i, s_i) \leftarrow \mathcal{A}(\sigma', Com', \pi'_0, \ldots, \pi'_{i-1}, s_{i-1})$ and gets a simulated proof $\pi'_i \leftarrow \mathsf{Sim}_2^D(\sigma', St_1, x_i)$.

The output of the experiment is $(\sigma', x_1, \pi'_1, \ldots, x_n, \pi'_n)$.

In the above, Sim_2^D is an oracle that is permitted to invoke a database oracle $D(.)$ and obtain values $D(x)$ for the keys x chosen by $\mathcal{A}$.