Concise Vector Commitments and their Applications to Zero-Knowledge Elementary Databases

Dario Catalano[1] and Dario Fiore[2⋆]

[1] Dipartimento di Matematica ed Informatica
Università di Catania, Italy.
catalano@dmi.unict.it
[2] École Normale Supérieure, CNRS - INRIA
Paris, France.
dario.fiore@ens.fr

Abstract. Zero knowledge sets (ZKS) [18] allow a party to commit to a secret set S and then to, non interactively, produce proofs for statements such as $x \in S$ or $x \notin S$. As recognized in [10], the key ingredient of ZKS is a primitive called (trapdoor) mercurial commitment.
This primitive was later generalized by Catalano, Fiore and Messina [9] who proposed the notion of (trapdoor) q-Mercurial Commitment (q-TMC for short) which allows to (mercurially) commit to vectors of q messages, rather than to single messages. This property is interesting as openings of specific vector positions can be made short and independent from q. Moreover, Catalano *et al.* show that one can get *Compact* Zero Knowledge Sets (i.e. ZKS with very short proofs) when combining q-TMC with a Merkle tree of branching degree q. The first construction of q-TMC with constant size openings, however, was proposed only very recently, by Libert and Yung in [16]. Their solution relies on the q-Diffie-Hellman Exponent assumption over bilinear groups.
In this paper we propose a general construction of q-TMC from a very simple primitive that we call *vector commitment*. Informally, a vector commitment is something that allows to commit to a vector of q elements in a way such that an adversary cannot open the commitment to two different messages at the same position.
Next, we show how to realize vector commitments with constant size openings from standard and well established assumptions such as RSA and CDH. When applied to the setting of Zero Knowledge Sets, our techniques directly imply simple and efficient solutions to several important open problems in the area. In particular we show the first (to the best of our knowledge) constructions of compact ZKS under standard assumptions (i.e. RSA and CDH) and the first known construction of compact and *Updatable* ZKS.

1 Introduction

Zero Knowledge Sets are a fascinating primitive that allows a party P, called the *prover*, to commit to a secret set S in a way such that he can later produce proofs for statements of the form $x \in S$ or $x \notin S$. The required properties are the following. First, any user V (the *verifier*) should be able to check the validity of the received proofs without learning any information on S (not even its size) beyond the mere membership (or non-membership) of the queried elements. Second, the produced proofs should be reliable in the sense that no dishonest prover should be able to convince V of the validity of a false statement. Zero Knowledge Sets (ZKS) were introduced by Micali, Rabin and Kilian in [18]. Their construction works in the so called *common reference string* (CRS) model, where (an honestly generated) random string is available to all parties, and relies on the intractability of the discrete logarithm problem[1].

⋆ Work partially done while student at University of Catania.

[1] More precisely, Micali *et al.* addressed the problem for the more general case of elementary databases (EDB) where each key x has associated a value $D(x)$ in the committed database. In the rest of this paper we will slightly abuse the notation and use the two acronyms ZKS and ZK EDB interchangeably to indicate the same primitive.

Later Chase *et al.* [10] abstracted away the construction of Micali *et al.* by introducing the notion of *mercurial commitment* and showing that ZKS can be built using this primitive (and collision resistant hash functions) as basic building block.

Mercurial commitments are "special" commitments where the binding property is somewhat relaxed to allow for two decommitting procedures: a *hard* and a *soft* one. In the committing phase one can decide as whether to create a hard commitment or a soft one. A hard commitment is like a standard one: it is created to a specific message m and it can be opened only w.r.t. m. Instead a soft commitment is initially created to "no message" and it can later be soft-opened (or *teased*) to any m, but it cannot be hard-opened. Chase *et al.* [10] show how to implement mercurial commitments from a variety of assumptions (either number theoretic or general)[2].

This result was further improved by Catalano, Dodis and Visconti [8] who show generic constructions of (trapdoor) mercurial commitments from the sole assumptions that one-way functions exist. From a theoretical point of view this latter result completed our understanding of zero knowledge sets, as it proved that collision-resistant hash functions are necessary and sufficient to construct ZKS in the CRS model.

From a practical perspective, however, things are much less exciting as none of the constructions [18, 10, 8] can be considered efficient enough for practical applications.

This is because, all constructions mentioned above are based on the following idea. To commit to a set $S \subset \{0,1\}^k$ one constructs an authenticated Merkle tree of depth k, where each internal node is filled with a mercurial commitment (rather than the hash) of its two children. A proof that $x \in \{0,1\}^k$ is in the committed set consists of the openings of all the commitments in the path from the root to the leaf labeled by x (more details about this construction can be found in [18, 10]). This implies that proofs have size linear in the height k of the tree. Now since 2^k is an upper bound for $|S|$, to guarantee that no information about $|S|$ is revealed, k has to be chosen so that 2^k is much larger than any reasonable set size.

The problem of constructing ZKS with shorter proofs was addressed by Catalano, Fiore and Messina in [9]. There they proposed a construction of ZKS that uses q-ary trees, instead of binary ones, and suggested a (new) notion of mercurial commitment that allows to implement it. Specifically, they introduced the notion of *q-mercurial commitment* and proposed a realization based on the Strong Diffie-Hellman assumption [4]. In a nutshell, a (trapdoor) q-mercurial commitment (qTMC for brevity) is a mercurial commitment that is created to an ordered sequence of exactly q messages and that can be opened to a single message of the sequence at a specified position. The nice feature of qTMC is that, if the length of each opening is short enough and independent of the branching factor of the tree, one can get ZKS having much shorter proofs than previous construction.

The "compact" construction of Catalano *et al.* significantly improved over previous ZK EDB schemes, but it cannot be considered fully satisfactory as the underlying qTMC is not as compact as one might desire in practice. The problem is that while the size of soft openings is independent of q, hard openings grow linearly in q. This results in an "unbalanced" ZK EDB construction where proofs of membership are much longer than proofs of non membership.

More recently, Libert and Yung [16] proposed a very elegant solution to this problem. Specifically, they managed to construct a q-mercurial commitment (that they called *concise*) achieving

[2] More precisely, Chase *et al.* show how to construct ZKS from *trapdoor* mercurial commitments. In a nutshell, this means that the scheme comes with a trapdoor that, if known, allows to violate the binding property of the commitment

constant-size (soft and hard) openings. This resulted in ZK EDB with very short proofs, as by increasing q one can get an arbitrarily "flat" tree[3]. Similarly to [9], the scheme of Libert and Yung [16] also relies on an asymptotic assumption in bilinear groups: the q-Diffie-Hellman Exponent [5].

1.1 Our contribution

In this paper we introduce a new technical tool called *vector commitment*. In a nutshell, a vector commitment (VC) allows to commit to an ordered sequence of q values (i.e. a vector) in a way such that it is later possible to open the commitment only w.r.t. to a specific position. More precisely, vector commitments are required to satisfy a *position binding* property which states that an adversary should not be able to open a commitment with two different values at the same position. Surprisingly, for the specific application of Zero Knowledge Sets, the position binding is all we need to prove our results. In particular, we show that vector commitments do not even need to be *hiding* if one wants to use them to construct ZKS. Furthermore, since in this work we are mostly interested in vector commitments with constant size openings, following [16], we call such schemes *concise*. So, we show the following theorem:

Theorem 1 (informal) *A (concise) trapdoor q-mercurial commitment can be obtained from a (concise) vector commitment and a trapdoor mercurial commitment.*

Having proved this, one can apply the result of Catalano *et al.* [9] to conclude that Compact ZKS can be built from mercurial commitments and concise vector commitments. Therefore, we turn our attention to the problem of realizing vector commitments and we show two realizations of VC from well established assumptions, namely RSA and Computational Diffie-Hellman[4].

Using the theorem above these can be combined with the RSA-based trapdoor Mercurial Commitment of Gennaro and Micali [12] (or, respectively, with the discrete log-based trapdoor Mercurial Commitment of Micali, Rabin and Kilian [18]) to get qTMCs with constant size (i.e. independent of q) hard and soft openings. Thus when instantiating the ZK EDB construction of Catalano *et al.* [9] with such schemes, one gets the first compact ZK-EDB realizations which are provably secure under standard assumptions.

In particular, our CDH realization induces proofs whose length is comparable to those induced by Libert and Yung's commitment [16], while relying on a more well established assumption.

Updatable ZK EDB with short proofs. As a second main contribution of this paper, we show what seems to be the first constructions of updatable ZKS with short proofs. The idea of updatable Zero Knowledge EDB was introduced by Liskov [17] to extend ZK EDB to the (very natural) case of "dynamic" databases. In an updatable ZK EDB the prover is allowed to change the value of some element x in the database and then output a new commitment C' and some update information U. Users holding a proof π_y for a $y \neq x$ valid w.r.t. C, should be able to use U to produce an updated proof π'_y that is valid w.r.t. C'. In [17] it is given a definition of *Updatable Zero Knowledge Databases* together with a construction based on mercurial commitments and Verifiable Random Functions [19] in the random oracle model. More precisely, Liskov introduced the notion of *updatable mercurial commitment* and proposed a construction, based on discrete logarithm, which is a variant of the mercurial commitment of Micali *et al.* [18].

[3] The only limitation is that the resulting common reference string grows linearly in q

[4] More precisely our second construction relies on the Inversion Computational Diffie-Hellman assumption (see Definition 3 in appendix A), which however has been shown to be equivalent to the standard CDH assumption [1]

In this paper, we provide the *first* realizations of "compact" Updatable Zero Knowledge EDB whose proofs and updates are much shorter than those of Liskov [17]. To do so we define and construct *updatable* concise vector commitments and we show how to use them to build Updatable ZK EDB from updatable qTMCs (which we also construct) and Verifiable Random Functions in the random oracle model. We stress that our solutions, apart from providing a solution to the open problem of realizing updatable ZK EDB with short proofs, further improve on previous work as they allow for much shorter updates as well[5].

As a byproduct of our results, we revisit the definition of security for updatable commitments. We propose a new (stronger) definition and a new updatable mercurial commitment realizing it. While the new definition has no impact on the construction of Updatable ZK EDB, it seems more natural than the one used so far and it might find applications in other contexts (see appendix K for details).

Other Contributions: Independent ZK EDB with short proofs. In [12] Gennaro and Micali addressed the problem of extending the security of zero knowledge databases to resist against adversaries that may correlate existing database commitments and proofs to create fake commitments or proofs. They introduced the notion of *Independent Zero Knowledge Databases* and proposed several realizations under different assumptions.

The construction of Gennaro and Micali uses multi-trapdoor mercurial commitments,[6] signature schemes and collision resistant hash functions as basic ingredients. Libert and Yung [16] adapted the notion of multi-trapdoor to the case of q-mercurial commitments and extended the construction of [12] to the case of q-ary trees. Moreover, they proposed an implementation of short Independent ZK EDB (based on the q-Diffie Hellman Exponent assumption). We further improve our understanding of this primitive by showing that a *multi-trapdoor* q-mercurial commitment can be easily obtained combining a vector commitment and a multi-trapdoor mercurial commitment. Moreover we show a concrete construction based on Strong RSA.

Other Applications of Vector Commitments. Even though for the specific application considered here Vector Commitments *do not* need to be hiding, they can be trivially extended to meet this requirement by simply composing them with standard commitments (i.e. one commits to a sequence of commitments). This simple extension further enlarges the range of applicability of our primitive to other scenarios as well. Indeed, since (concise) VC allow for constant size commitments and openings, they can be used to improve the bandwidth efficiency of cryptographic protocols. Here we briefly describe how to do this in the context of *pseudonymous credentials*, an application already identified by Kate *et al.* [15] when introducing polynomial commitments. Assume a user owns a set of credentials $(m_1, \ldots, m_q)$, each corresponding to a position i, and he wants to prove that his i-th credential is m_i (without revealing m_j, $\forall j \neq i$). The credentials are certified by a trusted entity who signs them. However, if the user makes a commitment C to $(m_1, \ldots, m_q)$ and the trusted entity signs C, then the user's proof can be just the signature on C and the opening of C to m_i at position i. If the latter proof has constant size, then the communication cost of the verification protocol is constant too. Kate *et al.* show how to solve this problem using polynomial commitment schemes. However, it is easy to see that our notion of concise VC fits this applica-

[5] This is because, in all known cosntructions, the size of the update information linearly depends on the height of the tree.

[6] Very informally, a multi-trapdoor mercurial commitment is defined like a standard one with the addition that commitments are created with respect to specific "tags". Moreover each tag t comes with an associate trapdoor key tk_t that allows to destroy the binding property of commitments created with respect to it.

tion as well. Moreover, our concrete schemes allow for two new solutions of this problem relying, respectively, on RSA and CDH (rather than on Strong DH in bilinear groups as in [15]).

Other Related Work. The notion of vector commitment is related with that of accumulator. Roughly speaking an accumulator is a commitment to a set of (at most) polynomially-many messages such that one can prove that an element is in the committed set (other notions allow to prove non-membership as well). For security an adversary should not be able to prove that an element is in the set while it is not. Many constructions of accumulators have been proposed so far [3, 2, 7, 20, 25] but none of them is required to meet security notions involving the position of messages[7].

The recent accumulator construction proposed by Camenisch *et al.* in [6] can be trivially modified to become a vector commitment according to our definition. In light of our results, this is not too surprising as that accumulator inspired the construction of the qTMC of Libert and Yung [16].

Ostrovsky, Rackoff and Smith [21] 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.

An elegant and efficient realization of ZKS from RSA was proposed by [26]. The main limitation of this construction though is that it does not generalize to elementary databases and it holds only in the random oracle model. More recently, Prabhakaran and Xue [22] 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.

Organization of the paper. The paper is organized as follows. In section 2 we recall the notion of trapdoor q-mercurial commitments and provide the relevant definitions for zero knowledge sets. In Section 3 we define vector commitments and we show how to build qTMC out of them. The impact of using vector commitments to build compact ZK EDBs is discussed in Section 4. Section 5 shows our constructions of Updatable ZK EDBs. In Section 6 we present our vector commitment from RSA. Finally, in section 7 we present the CDH based implementation. The construction of Independent ZK EDBs from strong RSA appears in appendix I

2 Preliminaries

In what follows we will denote with $k \in \mathbb{N}$ the security parameter. An algorithm $\mathcal{A}$ is said to be PPT if it is modeled as a probabilistic Turing machine that runs in time polynomial in k. Informally, we say that a function is *negligible* if it vanishes faster than the inverse of any polynomial. If S is a set, then $x \xleftarrow{\$} S$ indicates the process of selecting x uniformly at random over S (which in particular assumes that S can be sampled efficiently). For lack of space we give in appendix A the computational assumptions that we use in this paper.

[7] More precisely, the well known Merkle tree construction is a position binding vector commitment, but it does not have constant-size proofs

2.1 Trapdoor q-mercurial commitments

The notion of trapdoor q-mercurial commitment (qTMC) extends the notion of mercurial commitments (whose definition is given in appendix B) in the sense that it allows to (mercurially) commit to (ordered) sequences of messages, rather than to single ones. Formally qTMCs are defined in terms of the following algorithms (the definitions given in this section are taken from [9]):

$\mathsf{qKeyGen}(1^k, q)$ is a probabilistic algorithm that receives as input a security parameter k, the parameter q indicating the length of valid sequences and outputs a pair of public/private keys (pk, tk).

$\mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$ Given an ordered tuple of q messages, qHCom computes a hard commitment C to $(m_1, \cdots, m_q)$ using the public key pk and returns some auxiliary information 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 produces a hard decommitment π. 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 π proves that C is created to a sequence $(m_1, \cdots, m_q)$ such that $m_i = m$.

$\mathsf{qSCom}_{pk}()$ produces a soft commitment C and an auxiliary information 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 τ (also known as *"tease"*) to a message m at position i. The parameter $\mathsf{flag} \in \{\mathbb{H}, \mathbb{S}\}$ indicates if τ 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 a hard commitment and $m \neq m_i$.

$\mathsf{qSVer}_{pk}(m, i, C, \tau)$ checks if τ is a valid decommitment for C to m of index i. If it outputs 1 and τ 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 aux.

$\mathsf{qHEquiv}_{pk,tk}(m_1, \cdots, m_q, i, \mathsf{aux})$ The non-adaptive hard equivocation algorithm generates a hard decommitment π 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 τ to m of position i using the auxiliary information produced by the qFake algorithm.

Borrowing the terminology of [16], a trapdoor q-mercurial commitment is said to be *concise* if the size of the outputs of qHOpen and qSOpen is independent of q, and if an opening on m_i does not reveal information on the other messages m_j with $j \neq i$. For lack of space the correctness and security requirements for q-TMCs are given in appendix C

2.2 Zero-Knowledge Databases

Let D be a database and $[D]$ be the set of all the keys in 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 $D(x) = \perp$. A Zero Knowledge Elementary Database system is formally defined by a tuple of algorithms $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Prove}, \mathsf{V})$:

– $\mathsf{Setup}(1^k)$ takes as input the security parameter k and generates a common reference string CRS.
– $\mathsf{Commit}(CRS, D)$, 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.
– $\mathsf{Prove}(CRS, ZSK, x)$ On input the common reference string CRS, the secret key ZSK and an element x, the *prover* algorithm produces a proof π_x of either $D(x) = y$ or $D(x) = \perp$.
– $\mathsf{V}(CRS, ZPK, x, \pi_x)$ The *verifier* algorithm outputs y if $D(x) = y$, *out* if $D(x) = \perp$ or $\perp$ if the proof π_x is not valid.

We say that an EDB system is zero-knowledge if it satisfies *completeness*, *soundness* and *zero-knowledge*. The exact meaning of such requirements is given in appendix D. Here we only explain them informally. In a nutshell, *completeness* requires that proofs generated by honest provers are correctly verified; *soundness* imposes that a dishonest prover cannot prove false statements about elements of the database; finally *zero-knowledge* guarantees that proofs do not reveal any information on the database (beyond their validity).

3 Vector Commitments

In this section we introduce the notion of *vector commitment* and show how to use it to build trapdoor q-mercurial commitments, when used in conjunction with basic trapdoor mercurial commitments.

Informally speaking, a vector commitment allows to commit to an ordered sequence of values (i.e. a vector) in a way such that it is later possible to open the commitment only w.r.t. to a specific position. More formally, we define it via the following algorithms:

$\mathsf{VC.KeyGen}(1^k, q)$ The key generation algorithm takes in input the security parameter k and the size q of the committed vector (where q is at most $poly(k)$) and outputs a public key vpk.
$\mathsf{VC.Com}_{vpk}(m_1, \ldots, m_q)$ On input a sequence of q messages $m_1, \ldots, m_q$ and the public key vpk, the committing algorithm outputs a commitment string C and an auxiliary information $\mathtt{aux}$.
$\mathsf{VC.Open}_{vpk}(m, i, \mathtt{aux})$ Let $(C, \mathtt{aux}) \xleftarrow{\$} \mathsf{VC.Com}_{vpk}(m_1, \ldots, m_q)$. If $m = m_i$ the opening algorithm produces a proof Λ_i that m is the i-th committed message. Otherwise it outputs an error symbol $\perp$.
$\mathsf{VC.Ver}_{vpk}(C, m, i, \Lambda_i)$ The verification algorithm accepts (i.e. outputs 1) only if Λ_i is a valid proof that C was created to a sequence $m_1, \ldots, m_q$ such that $m = m_i$.

For correctness we require that for all honestly generated public keys $vpk \xleftarrow{\$} \mathsf{VC.KeyGen}(1^k, q)$ and $\forall (m_1, \ldots, m_q) \in \mathcal{M}^q$ the following statement holds with overwhelming probability $\forall i = 1, \ldots, q$:

$$(C, \mathtt{aux}) \xleftarrow{\$} \mathsf{VC.Com}_{vpk}(m_1, \ldots, m_q) : \mathsf{VC.Ver}_{vpk}(C, m_i, i, \mathsf{VC.Open}_{vpk}(m_i, i, \mathtt{aux})) = 1.$$

The attractive feature of vector commitments is that they are required to meet a very weak security requirement, that we call *position binding* property. Informally, this means that it should be infeasible, for a polynomially bounded adversary having knowledge of vpk, to come up with a VC C and two different openings for the same position i. More formally, we say that a vector commitment has position binding if $\forall i = 1, \ldots, q$ and for every PPT adversary $\mathcal{A}$ the following probability (which is taken over all honestly generated public keys) is at most negligible in the security parameter:

$$Pr[(C, m, m', i, \Lambda, \Lambda') \leftarrow \mathcal{A}(vpk) : \mathsf{VC.Ver}_{vpk}(C, m, i, \Lambda) = 1 \wedge \mathsf{VC.Ver}_{vpk}(C, m', i, \Lambda') = 1 \wedge m \neq m']$$

Moreover, we say that a vector commitment is *concise* if the size of the outputs of $\mathsf{VC.Open}$ is independent of q. We stress that Vector Commitments *are not* required to be hiding. As we will show below, position binding is the sole requirement needed to prove theorem 1. However, if one would like to have a VC which is also hiding, then it is sufficient to compose it with a standard commitment scheme.

3.1 Trapdoor q-Mercurial Commitments from Vector Commitments and Mercurial Commitments

Now we show how to combine (concise) vector commitments and standard trapdoor commitment to obtain (concise) trapdoor qTMC.

Let $\mathsf{TMC} = (\mathsf{KeyGen}, \mathsf{HCom}, \mathsf{HOpen}, \mathsf{HVer}, \mathsf{SCom}, \mathsf{SOpen}, \mathsf{SVer}, \mathsf{Fake}, \mathsf{HEquiv}, \mathsf{SEquiv})$ be a trapdoor mercurial commitment and $\mathsf{VC} = (\mathsf{VC.KeyGen}, \mathsf{VC.Com}, \mathsf{VC.Open}, \mathsf{VC.Ver})$ be a vector commitment. We construct a trapdoor qTMC as follows:

$\mathsf{qKeyGen}(1^k)$ Run $vpk \xleftarrow{\$} \mathsf{VC.KeyGen}(1^k, q)$ and $(PK_{\mathsf{TMC}}, TK_{\mathsf{TMC}}) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and set $pk = (vpk, PK_{\mathsf{TMC}})$ and $tk = TK_{\mathsf{TMC}}$.

$\mathsf{qHCom}_{pk}(m_1, \ldots, m_q)$ For $i = 1$ to q compute $(C_i, \mathsf{aux}^i_{\mathsf{TMC}}) \xleftarrow{\$} \mathsf{HCom}_{PK_{\mathsf{TMC}}}(m_i)$. Next, compute $(C, \mathsf{aux}_{\mathsf{VC}}) \leftarrow \mathsf{VC.Com}(C_1, \ldots C_q)$. The output is C and the auxiliary information is $\mathsf{aux} = (\mathsf{aux}_{\mathsf{VC}}, m_1, C_1, \mathsf{aux}^1_{\mathsf{TMC}}, \ldots, m_q, C_q, \mathsf{aux}^q_{\mathsf{TMC}})$.

$\mathsf{qHOpen}_{pk}(m_i, i, \mathsf{aux})$ Extract $(m_i, C_i, \mathsf{aux}^i_{\mathsf{TMC}})$ from aux and set $\Lambda_i \leftarrow \mathsf{VC.Open}_{vpk}(C_i, i, \mathsf{aux}_{\mathsf{VC}})$. The opening information is $\tau_i = (\Lambda_i, C_i, \pi_i)$ where π_i is the output of $\mathsf{HOpen}_{PK_{\mathsf{TMC}}}(m_i, \mathsf{aux}^i_{\mathsf{TMC}})$.

$\mathsf{qHVer}_{pk}(C, m_i, i, \tau_i)$ Parse τ_i as (Λ_i, C_i, π_i). The hard verification algorithm returns 1 if and only if both $\mathsf{HVer}_{PK_{\mathsf{TMC}}}(C_i, m_i, \pi_i)$ and $\mathsf{VC.Ver}_{vpk}(C, C_i, i, \Lambda_i)$ return 1.

$\mathsf{qSCom}_{pk}()$ For $i = 1$ to q compute $(C_i, \mathsf{aux}^i_{\mathsf{TMC}}) \leftarrow \mathsf{SCom}_{PK_{\mathsf{TMC}}}()$. Next, compute $(C, \mathsf{aux}_{\mathsf{VC}}) \leftarrow \mathsf{VC.Com}(C_1, \ldots C_q)$. The output is C and the auxiliary information is $\mathsf{aux} = (\mathsf{aux}_{\mathsf{VC}}, m_1, C_1, \mathsf{aux}^1_{\mathsf{TMC}}, \ldots, m_q, C_q, \mathsf{aux}^q_{\mathsf{TMC}})$.

$\mathsf{qSOpen}_{pk}(m_i, i, \mathsf{flag}, \mathsf{aux})$ Extract $(m_i, C_i, \mathsf{aux}^i_{\mathsf{TMC}})$ from aux and set $\Lambda_i \leftarrow \mathsf{VC.Open}_{vpk}(C_i, i, \mathsf{aux}_{\mathsf{VC}})$. The opening information is $\tau_i = (\Lambda_i, C_i, \pi_i)$ where π_i is the output of $\mathsf{SOpen}_{PK_{\mathsf{TMC}}}(m_i, \mathsf{aux}^i_{\mathsf{TMC}})$.

$\mathsf{qSVer}_{pk}(C, m, i, \tau_i)$ Parse τ_i as (Λ_i, C_i, π_i). The soft verification algorithm returns 1 if and only if both $\mathsf{SVer}_{PK_{\mathsf{TMC}}}(C_i, m_i, \pi_i)$ and $\mathsf{VC.Ver}_{vpk}(C, C_i, i, \Lambda_i)$ return 1.

$\mathsf{qFake}_{pk,tk}()$ This is the same as the qSCom algorithm.

$\mathsf{qHEquiv}_{pk,tk}(m, i, \mathsf{aux})$ Extract $(C_i, \mathsf{aux}^i_{\mathsf{TMC}})$ (for all $i = 1$ to q) and set $\Lambda_i \leftarrow \mathsf{VC.Open}_{vpk}(C_i, i, \mathsf{aux}_{\mathsf{VC}})$. The hard equivocation is $\tau_i = (\Lambda_i, C_i, \pi_i)$ where π_i is the output of $\mathsf{HEquiv}_{PK_{\mathsf{TMC}}, tk_{\mathsf{TMC}}}(m, \mathsf{aux}^i_{\mathsf{TMC}})$

$\mathsf{qSEquiv}_{pk,tk}(m, i, \mathsf{aux})$ Extract $(C_i, \mathsf{aux}^i_{\mathsf{TMC}})$ (for all $i = 1$ to q) and set $\Lambda_i \leftarrow \mathsf{VC.Open}_{vpk}(C_i, i, \mathsf{aux}_{\mathsf{VC}})$. The soft equivocation is $\tau_i = (\Lambda_i, C_i, \pi_i)$ where π_i is the output of $\mathsf{SEquiv}_{PK_{\mathsf{TMC}}, tk_{\mathsf{TMC}}}(m, \mathsf{aux}^i_{\mathsf{TMC}})$

The correctness of the scheme easily follows from the correctness of the underlying tools. We prove its security via the following theorem (whose proof is given in appendix E.1) :

Theorem 1. *Assuming that* TMC *is a trapdoor mercurial commitment and* VC *is a vector commitment, then the scheme defined above is a trapdoor q-mercurial commitment.*

4 Concise Vector Commitments and Compact ZKS

In [9] it is shown how to use qTMCs to build ZK EDBs whose proofs are (potentially) much shorter than those obtained when adopting basic (trapdoor) mercurial commitments as underlying building

block. Here we first, briefly, recall the construction by Catalano *et al.* [9] and then we discuss the impact on it when using (concise) vector commitments.

To commit to a database D the committer proceeds as follows. Each key x is associated to a leaf in a q-ary tree of height h such that $q^h = 2^k$ where k is some appropriate security parameter. Thus x can be viewed as a labeling, in a q-ary encoding, for a unique leaf in the tree. Since the number of all possible keys is exponential in k, the committing phase is made efficient by pruning those subtrees whose leaves correspond to keys that are *not* in the database. Only the roots (called "frontier nodes") of these subtrees are kept in memory and "filled" with soft commitments. The remaining nodes are filled as follows. Each leaf contains a hard commitment (computed using a standard trapdoor mercurial commitment scheme) of the value $D(x)$ (or 0 if $D(x) = \perp$). Each internal (non frontier) node contains the hard q-commitment of the values contained in its q children. The commitment contained in the root of the tree is the (public) commitment of the entire database.

To prove that $x \in D$ (for instance that $D(x) = y$), the prover 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, each hard q-commitment is opened 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 softly opened to the values contained in theirs newly generated children. For completeness we give a detailed description of this construction in appendix D.1.

Thus, combining the results from [9] with theorem 1 it is clear that to construct compact ZKS it is sufficient to realize concise implementations of vector commitments. In the next sections we do this from two well established assumptions, namely RSA and Computational Diffie-Hellman. When using these constructions in conjunction with (trapdoor) TMCs relying on the same (or weaker) assumptions (such as [12] or [18]), one gets *compact* realizations of ZKS from RSA and CDH respectively.

5 Updatable Zero-Knowledge Databases with short proofs and updates

In most practical applications databases are frequently updated. The constructions of ZK EDBs described so far do not deal with this and the only way of updating a ZK EDB is to actually recompute the entire commitment from scratch every time the database changes. This is highly undesirable as previously issued proofs can no longer be valid.

This problem was studied by Liskov in [17] where he showed how to build Updatable ZK EDB by appropriately modifying the basic approach of combining Merkle trees and mercurial commitments. In particular, rather than using standard mercurial commitments, Liskov employed a new primitive called *updatable mercurial commitment*. Very informally, updatable mercurial commitments are like standard ones with the additional feature that they allow for two update procedures. The committer can change the message inside the commitment and produce: a new commitment and an update information. These can later be used by verifiers to update their commitments and the associated proofs (that will be valid w.r.t. the new commitment). Therefore whenever the prover changes some value $D(x)$ in the database, first he has to update the commitment in the leaf labeled by x and then he updates all the commitments in the path from x to the root. The new database commitment is the updated commitment in the root node while the database update information contains the update informations for all the nodes involved in the update.

A natural question raised by the methodology above is if the zero-knowledge property remains preserved after an update occurs, as the latter reveals information about the updated key. To solve this issue Liskov proposed to "mask" the label of each key x (i.e. the paths in the tree) using a pseudorandom pseudonym $N(x)$ and he relaxed the zero-knowledge property to hold w.r.t. $N()$. Further details can be found in [17].

In [17] two constructions of updatable mercurial commitments are given. One is generic and uses both standard and mercurial commitments. The other one is direct and builds from the DL-based mercurial commitment of [18].

Here we show that concise updatable q-mercurial commitments can be easily constructed using vector commitments. In particular we show that if the vector commitment is updatable (according to a definition that we give below), then it can be combined with an updatable mercurial commitment to build an updatable q-mercurial commitment. Then, if we apply the methodology of Liskov sketched above, adapted with the construction of Catalano *et al.* [9], we build the first *Compact* Updatable ZK EDB. Interestingly, the resulting scheme improves over the construction in [17] both in terms of proofs' size and in terms of length of update informations, as both grow linearly in $log_q 2^k$ (which is strictly less than k for $q > 2$). For lack of space, more details about updatable mercurial commitments are deferred to appendix H.

Updatable q-mercurial commitments An updatable (concise) q-mercurial commitment is defined like a standard one with the following two additional algorithms:

$\mathsf{qUpdate}_{pk}(S, \mathsf{aux}, m', i)$ This algorithm is run by the committer who produced S (and holds the corresponding aux) and wants to change the i-th commited message with m'. The algorithm takes as input m' and the position i and outputs a new commitment S' and an update information U.

$\mathsf{qProofUpdate}_{pk}(S, \tau_j, m', i, U)$ This algorithm can be run by any user who holds a proof τ_j for some message at position j in S and allows the user to produce a new proof τ'_j (and the updated commitment S') which will be valid w.r.t. S' that contains m' as the new message at position i. The value U contains the update information which is needed to compute such values.

The q-mercurial binding property is defined as usual, namely for any PPT adversary it is computationally infeasible to open a commitment (even an updated one) to two different messages at the same position.

Hiding and equivocations for updatable q-mercurial commitments easily follow from those of updatable mercurial commitments given in the previous section by extending them to the case of sequences of q messages.

Updatable Vector Commitments An updatable (concise) vector commitment is defined like a standard one with the following two additional algorithms:

$\mathsf{vUpdate}_{pk}(C, \mathsf{aux}, m', i)$ This algorithm is run by the committer who produced C (and holds the corresponding aux) and wants to change the i-th commited message with m'. The algorithm takes as input m' and the position i and outputs a new commitment C' and an update information U.

$\mathsf{vProofUpdate}_{pk}(C, \Lambda_j, m', i, U)$ This algorithm can be run by any user who holds a proof Λ_j for some message at position j in C and allows the user to produce a new proof Λ'_j (and the

updated commitment C') which will be valid w.r.t. C' that contains m' as the new message at position i. The value U contains the update information which is needed to compute such values.

The position binding property remains the same, namely for any PPT adversary it is computationally infeasible to open a commitment (even an updated one) to two different messages at the same position.

5.1 A Construction of Updatable concise q-mercurial commitments

In this section we extend the result of of Theorem 1 to the updatable case. Namely we show that an updatable concise q-mercurial commitment can be built using an updatable mercurial commitment uTMC and an updatable vector commitment uVC. The construction is essentially the same as that given in section 3.1 augmented with the following update algorithms:

$\mathsf{qUpdate}_{pk}(C, \mathsf{aux}, m', i)$ Parse aux as $(m_1, C_1, \mathsf{aux}^1_{\mathsf{uTMC}}, \ldots, m_q, C_q, \mathsf{aux}^q_{\mathsf{uTMC}})$, extract $(C_i, m_i, \mathsf{aux}^i_{\mathsf{uTMC}})$ from it and run $(C'_i, U_{\mathsf{uTMC}}) \leftarrow \mathsf{Update}_{pk_{\mathsf{uTMC}}}(C_i, \mathsf{aux}^i_{\mathsf{uTMC}}, m')$. Then update the commitment $(C', U') \leftarrow \mathsf{vUpdate}_{vpk}(C, \mathsf{aux}, C'_i, i)$ and output C' and $U = (U', C_i, C'_i, i, U^i_{\mathsf{uTMC}})$.

$\mathsf{qProofUpdate}_{pk}(C, \tau_j, m', i, U)$ Who holds a proof $\tau_j = (\Lambda_j, C_j, \pi_j)$ that is valid w.r.t. to C for some message at position j, can use the update information U to compute the updated commitment C' and produce a new proof τ'_j which will be valid w.r.t. the new C'. We distinguish two cases:

1. $i \neq j$. Compute $(C', \Lambda'_j) \leftarrow \mathsf{vProofUpdate}_{vpk}(C, \Lambda_j, C'_i, i, U')$ and output the updated commitment C' and the updated proof $\tau_j = (\Lambda'_j, C_j, \pi_j)$.
2. $i = j$. Let $(C'_i, \pi'_i) \leftarrow \mathsf{ProofUpdate}_{pk_{\mathsf{uTMC}}}(C_i, m', \pi_i, U^i_{\mathsf{uTMC}})$. Compute the updated commitment as $(C', \Lambda'_i) \leftarrow \mathsf{vProofUpdate}_{vpk}(C, \Lambda_i, C'_i, i, U')$ and the updated proof is $\tau'_i = (\Lambda'_i, C'_i, \pi'_i)$.

Theorem 2. *If* uTMC *is an updatable trapdoor mercurial commitment and* uVC *is an updatable vector commitment, then the scheme given above is an updatable concise trapdoor q-mercurial commitment.*

Proof (Sketch). The proof of the q-mercurial binding remains the same as that of Theorem 1 and the same holds for hiding and equivocations of "standard" commitments. To see that hiding and equivocations hold even for updated commitments, consider the update information $U = (U', C_i, C'_i, i, U^i_{\mathsf{uTMC}})$ where C'_i is an updated commitment to m'. Then it is easy to see that U does not reveal information about m' as long as the scheme uTMC is hiding.

6 A Concise Vector Commitment from RSA

Here we propose an implementation of concise vector commitments based on the RSA assumption (for its definition see appendix A). Combined with the theorems given above, this leads (when combined with the TMC of Gennaro and Micali [12], recalled in appendix F) to a constructions of qTMC solely relying on the RSA assumption.

$\mathsf{VC.KeyGen}(k, \ell, q)$ Randomly choose two $k/2$-bit primes p_1, p_2 and set $N = p_1 p_2$ and q $(\ell+1)$-bit primes $e_1, \ldots, e_q$ not dividing $\phi(N)$. Moreover, let $H : \{0,1\}^* \leftarrow \{0,1\}^\ell$ be a collision resistant hash function (chosen at random from a collision resistant function family $\mathcal{H}$) and a be a random (high order) element in $\mathbb{Z}_N$. For $i = 1$ to q set

$$S_i = a^{\prod_{j=1, j\neq i}^{q} e_j}$$

The public key vpk is set as $(N, a, S_1, \ldots, S_q, H, e_1, \ldots, e_q)$.

$\mathsf{VC.Com}_{vpk}(m_1, \ldots, m_q)$ Set $c_i \leftarrow H(m_i)$ for all $i = 1 \ldots q$ and compute

$$C \leftarrow S_1^{c_1} S_2^{c_2} \cdots S_q^{c_q}$$

The output is C and the auxiliary information is $\mathsf{aux} = (m_1, \ldots, m_q)$.
$\mathsf{VC.Open}_{vpk}(m_i, i, \mathsf{aux})$, Output

$$\Lambda_i \leftarrow \sqrt[e_i]{\prod_{j=1, j\neq i}^{q} S_j^{H(m_j)}} \bmod N$$

Notice that knowledge of vpk allows to compute Λ_i efficiently *without* the factorization of N.
$\mathsf{VC.Ver}_{pk}(C, m_i, i, \Lambda_i)$ The verification algorithm returns 1 if

$$C = S_i^{H(m_i)} \Lambda_i^{e_i} \bmod N$$

In order for the verification process to be correct, notice that one should also verify (only once) the validity of the public key by checking that the S_i's are correctly generated with respect to a and the exponents $e_1, \ldots, e_q$.

The correctness of the scheme can be easily verified by inspection. We prove the remaining properties via the following theorem whose actual proof appears in appendix E.2.

Theorem 3. *Assuming that $\mathcal{H}$ is a family of collision resistant hash functions and that the RSA assumption holds, then the scheme defined above is a concise vector commitment.*

Efficiency into building ZKS According to the construction of [9] (see Section 4), a proof of membership contains one commitment and its hard opening for each internal level of the tree plus the hard opening of the root commitment. On the other hand, a proof of non-membership requires to eventually reconstruct the missing part of the tree and to provide one commitment and its soft opening for each level of the tree, plus the soft opening of the root commitment. When using the qTMC obtained from the vector commitment given above and the RSA-based TMC of [12] a proof of membership requires $6h - 1$ elements of $\mathbb{Z}_N$ while a proof of non-membership requires $5h - 1$ elements[8].

Variant construction In appendix J we present a variant of the scheme given above that allows for a constant size public key. Although the algorithms still run in time linear in q, this is the first example of a scheme that can obtain a constant size public key.

6.1 Updatable vector commitment from RSA

The Vector Commitment given above can be made updatable by adding the following algorithms.

$\mathsf{vUpdate}_{pk}(C, \mathsf{aux}, m', i)$ Parse aux as $(m_1, \ldots, m_q)$. Recompute the commitment $C' = (\prod_{j=1, j\neq i}^{q} S_j^{c_j}) S_i^{c_i'}$ where $c_i' = H(m_i')$. Finally output C' and $U = (m_i, m_i', i)$.

[8] These numbers already take into account some simple efficiency improvements allowed by our construction. For instance, we don't need to require additional commitments for the leaves of the tree.

$\mathsf{vProofUpdate}_{pk}(C, \Lambda_j, m', i, U)$ Who holds a proof Λ_j, that is valid w.r.t. to C for some message at position j, can use the update information U to compute the updated commitment C' and produce a new proof Λ'_j which will be valid w.r.t. the new C'. We distinguish two cases:

1. $i \neq j$. Let c_i and c'_i be $H(m_i)$ and $H(m'_i)$ respectively. Then compute the updated commitment as $C' = CS_i^{c'_i - c_i}$ while the updated proof is $\Lambda'_j = \Lambda_j \sqrt[e_j]{S_i^{c'_i - c_i}}$ (notice that this e_j-th root can be efficiently computed using the elements in the public key).
2. $i = j$. Compute the updated commitment as $C' = CS_i^{c'_i - c_i}$ while the updated proof remains the same Λ_i.

Theorem 4. *If the RSA assumption holds and H is collision resistant, then the scheme given above is an updatable concise vector commitment.*

It is easy to see that the proof of the position binding remains the same as that of Theorem 3.

Now combining the VC above with an updatable variant of Gennaro and Micali [12] basic mercurial commitment (whose descrition, for lack of space, is postponed to appendix H.1), theorem 2 brings us an RSA-based construction of updatable qTMCs, which, in turn, leads to the first (to the best of our knowledge) realization of compact and updatable ZKS.

7 A Concise Vector Commitment from CDH

Here we propose an implementation of concise vector commitments based on the CDH assumption. Precisely, the security of the scheme reduces to the Inversion Computational Diffie-Hellman assumption (see Definition 3 in appendix A). However, this has been shown to be equivalent to the standard CDH assumption [1].

$\mathsf{KeyGen}(k, q)$ Let $\mathbb{G}, \mathbb{G}_T$ be two bilinear groups of order p equipped with a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let $g \in \mathbb{G}$ be a random generator. Randomly choose $z_1, \ldots, z_q \xleftarrow{\$} \mathbb{Z}_p$. For all $i = 1, \ldots, q$ set: $g_i = g^{z_i}, h_i = g^{\prod_{k \neq i} z_k}$. For all $i, j = 1, \ldots, q$, $i \neq j$ set $h_{i,j} = g^{\prod_{k \neq i,j} z_k} = h_j^{1/z_i}$.

Set $pk = (g, g_1, \ldots, g_q, \{h_i\}_{i \in [q]}, \{h_{i,j}\}_{i,j \in [q], i \neq j})$.

$\mathsf{Com}_{pk}(m_1, \ldots, m_q)$ Compute

$$C = h_1^{m_1} h_2^{m_2} \cdots h_q^{m_q}$$

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

$\mathsf{Open}_{pk}(m_i, i, \mathsf{aux})$, Set

$$\Lambda_i = \Big(\prod_{j=1, j \neq i}^{q} h_j^{m_j}\Big)^{1/(z_i)} = \prod_{j=1, j \neq i}^{q} h_{i,j}^{m_j}$$

$\mathsf{Ver}_{pk}(C, m_i, i, \Lambda_i)$ Check that

$$e(C \cdot h_i^{-m_i}, g) = e(\Lambda_i, g_i)$$

The correctness of the scheme can be easily verified by inspection. We prove the remaining properties via the following theorem (see appendix E.3 for the actual proof).

Theorem 5. *If the CDH assumption holds, then the scheme defined above is a concise vector commitment.*

Efficiency into building ZKS We consider the size of proofs of a ZK EDB built using the qTMC instantiated from the vector commitment given in this section and the TMC of Micali-Rabin-Kilian based on Discrete Log [18].

To improve the space-efficiency of the scheme it is convenient to implement our VC using asymmetric pairings. In this case the pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is defined over groups $\mathbb{G}_1$ and $\mathbb{G}_2$ constructed as follows. $\mathbb{G}_1$ is realized as a subgroup of points of an elliptic curve E over a finite field $\mathbb{F}_p$ of size p, where p is an ℓ bits prime. If e is a parameter called *embedding degree*, $\mathbb{G}_2$ is a subgroup of $E(\mathbb{F}_{p^e})$ and $\mathbb{G}_T \subset E(\mathbb{F}^*_{p^e})$. In particular we consider elliptic curves with embedding degree $e = 12$ and CM discriminant $D = -3$. As suggested in [23], for the case of *Type 3 groups* (see [23] for details), such parameters enable to obtain elements of $\mathbb{G}_2$ that have size twice the size of elements of $\mathbb{G}_1$.

In this case a proof of membership requires $6h - 1$ group elements while a proof of non-membership requires $5h - 1$ elements[9]

In the scheme in [16] proofs of membership and non-membership require $5h + 5$ and $4h + 4$ group elements respectively. So our scheme is slightly less efficient and requires a longer public key (i.e. common reference string), but relies on a much more established assumption (namely CDH vs q-Diffie-Hellman exponent).

7.1 Updatable vector commitment from CDH

The Vector Commitment given above can be made updatable by adding the following algorithms.

$\mathsf{vUpdate}_{pk}(C, \mathsf{aux}, m', i)$ Parse aux as $(m_1, \ldots, m_q)$. Recompute the commitment $C' = (\prod_{j=1, j\neq i}^{q} h_j^{m_j}) h_i^{m'_i}$. Finally output C' and $U = (m_i, m'_i, i)$.

$\mathsf{vProofUpdate}_{pk}(C, \Lambda_j, m', i, U)$ Who holds a proof Λ_j, that is valid w.r.t. to C for some message at position j, can use the update information U to compute the updated commitment C' and produce a new proof Λ'_j which will be valid w.r.t. the new C'. We distinguish two cases:

1. $i \neq j$. Compute the updated commitment as $C' = C h_i^{m'_i - m_i}$ while the updated proof is $\Lambda'_j = \Lambda_j (h_i^{m'_i - m_i})^{1/z_j} = \Lambda_j h_{j,i}^{m'_i - m_i}$.
2. $i = j$. Compute the updated commitment as $C' = C h_i^{m'_i - m_i}$ while the updated proof remains the same Λ_i.

Theorem 6. *If the CDH assumption holds, then the scheme given above is an updatable concise vector commitment.*

It is easy to see that the proof of the position binding remains the same as that of Theorem 5.

Now, if we combine the VC given above with the updatable mercurial commitment described by Liskov in [17] (which is based on discrete-log), then theorem 2 brings us a CDH-based construction of updatable qTMCs, which, in turn, leads to another realization of compact and updatable ZKS. This latter construction is particularly interesting, as when implementing it over suitable elliptic curves it allows to adopt very short parameters and to significantly improve over the scheme in [17]. In particular, our solution achieves the same improvement (in terms of length of the proofs) as the as that obtained by Libert and Yung's scheme w.r.t. the Micali, Rabin and Kilian's construction, namely proofs that are about 70% shorter.

[9] Again, these numbers already take into account some simple efficiency improvements that are allowed by our construction.

8 Conclusions

In this paper we introduced the notion of Vector Commitment, a simple but powerful primitive that can be used to realize compact and updatable Zero Knowledge Sets from standard and well established assumptions such as RSA and CDH. To the best of our knowledge no the ours is the first realization of compact and updatable ZKS. Moreover, all previously known constructions of basic (i.e. non updatable) compact ZKS needed to rely on young asymptotic assumptions on bilinear groups, such as q-SDH or q-DHE. Finally, as a nice add-on, our methodology allows also to construct Independent Zero Knowledge Databases with short proofs from the Strong RSA assumption (see section I for details on this).

References

1. Feng Bao, Robert Deng, and HuaFei Zhu. Variations of diffie-hellman problem. In Sihan Qing, Dieter Gollmann, and Jianying Zhou, editors, *Information and Communications Security*, volume 2836 of *Lecture Notes in Computer Science*, pages 301–312. Springer Berlin / Heidelberg, 2003.
2. Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494, Konstanz, Germany, May 11–15, 1997. Springer, Berlin, Germany.
3. Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285, Lofthus, Norway, May 23–27, 1993. Springer, Berlin, Germany.
4. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.
5. Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany.
6. Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 481–500, Irvine, CA, USA, March 18–20, 2009. Springer, Berlin, Germany.
7. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.
8. Dario Catalano, Yevgeniy Dodis, and Ivan Visconti. Mercurial commitments: Minimal assumptions and efficient constructions. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 120–144, New York, NY, USA, March 4–7, 2006. Springer, Berlin, Germany.
9. Dario Catalano, Dario Fiore, and Mariagrazia Messina. Zero-knowledge sets with short proofs. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 433–450, Istanbul, Turkey, April 13–17, 2008. Springer, Berlin, Germany.
10. Melissa Chase, Alexander Healy, Anna Lysyanskaya, Tal Malkin, and Leonid Reyzin. Mercurial commitments with applications to zero-knowledge sets. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 422–439, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.
11. Rosario Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 220–236, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.
12. Rosario Gennaro and Silvio Micali. Independent zero-knowledge sets. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 34–45, Venice, Italy, July 10–14, 2006. Springer, Berlin, Germany.

13. J. Hastad, A. Schrift, and A. Shamir. The discrete logarithm modulo a composite hides o(n) bits. In *Journal of Computer and System Science*, volume 47 (3), pages 376–404, 1993.
14. Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, Lecture Notes in Computer Science, pages 654–670, Santa Barbara, CA, USA, August 2009. Springer, Berlin, Germany.
15. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010*, volume 6477, page 178, 2010.
16. Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *TCC 2010: 7th Theory of Cryptography Conference*, Lecture Notes in Computer Science, pages 499–517. Springer, Berlin, Germany, 2010.
17. Moses Liskov. Updatable zero-knowledge databases. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 174–198, Chennai, India, December 4–8, 2005. Springer, Berlin, Germany.
18. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *44th Annual Symposium on Foundations of Computer Science*, pages 80–91, Cambridge, Massachusetts, USA, October 11–14, 2003. IEEE Computer Society Press.
19. Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 120–130, New York, New York, USA, October 17–19, 1999. IEEE Computer Society Press.
20. Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292, San Francisco, CA, USA, February 14–18, 2005. Springer, Berlin, Germany.
21. Rafail Ostrovsky, Charles Rackoff, and Adam Smith. Efficient consistency proofs for generalized queries on a committed database. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *ICALP 2004: 31st International Colloquium on Automata, Languages and Programming*, volume 3142 of *Lecture Notes in Computer Science*, pages 1041–1053, Turku, Finland, July 12–16, 2004. Springer, Berlin, Germany.
22. Manoj Prabhakaran and Rui Xue. Statistically hiding sets. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 100–116, San Francisco, CA, USA, April 20–24, 2009. Springer, Berlin, Germany.
23. K.G. Paterson S.D. Galbraith and N.P. Smart. Pairings for cryptographers, Cryptology ePrint Archive, Report 2006/165 2006. http://eprint.iacr.org.
24. Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. Comput. Syst.*, 1(1):38–44, 1983.
25. Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. A new dynamic accumulator for batch updates. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS 07: 9th International Conference on Information and Communication Security*, volume 4861 of *Lecture Notes in Computer Science*, pages 98–112, Zhengzhou, China, December 12–15, 2007. Springer, Berlin, Germany.
26. Rui Xue, Ning-Hui Li, and Jiang-Tao Li. Algebraic construction for zero-knowledge sets. *Journal of Computer Science and Technology*, 23(2):166–175.

A Computational assumptions

An integer N is called *RSA modulus* if it is the product of two distinct prime numbers p, q. Given a public RSA modulus N, a public exponent e, such that $\gcd(e, \phi(N)) = 1$, and a random value $z \in \mathbb{Z}_N$, the RSA problem asks to compute the unique $y \in \mathbb{Z}_N$ such that $z \equiv y^e \bmod N$. The public exponent can be chosen according to various distributions, in particular different distributions give rise to different variant of the problem. In this paper we consider the RSA problem where e is chosen as a random $\ell + 1$ bit prime (for some parameter ℓ). More formally, the corresponding RSA assumption states the following.

Assumption 1 (RSA) *Let $k \in \mathbb{N}$ be the security parameter, N a random RSA modulus of length k, z be a random element in $\mathbb{Z}_N$ and e be an $(\ell+1)$-bit prime (for a parameter ℓ).Then we say that the RSA assumption holds if for any PPT adversary $\mathcal{A}$ the probability*

$$Pr[y \leftarrow \mathcal{A}(N, e, z) : y^e = z \bmod N]$$

is negligible in k.

The Strong RSA Assumption was introduced by Baric and Pfitzmann in [2]. Essentially it is a variant of RSA where the adversary is allowed to choose the exponent e ($e \neq 1$) for which it has to extract the root.

Assumption 2 (Strong-RSA) *Let N be a random RSA modulus of length k where $k \in \mathbb{N}$ is the security parameter and z be a random element in $\mathbb{Z}_N$. Then we say that the Strong-RSA assumption holds if for any PPT adversary $\mathcal{A}$ the probability*

$$Pr[(y, e) \leftarrow \mathcal{A}(N, z) : y^e = z \bmod N \wedge e \neq 1]$$

is negligible in k.

Assumption 3 (Inv-CDH) *Let $k \in \mathbb{N}$ be the security parameter. Let $\mathbb{G}$ be a group of prime order p, $g \in \mathbb{G}$ be a generator and $a \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. We say that the Inversion Computational Diffie-Hellman Assumption holds in $\mathbb{G}$ if for every PPT algorithm $\mathcal{A}$, the probability $\mathsf{P}_2[\mathcal{A}(g, g^a) = g^{1/a}]$ is at most a negligible function in k.*

In [1], it is shown that the Inv-CDH assumption is equivalent to the classical Computational Diffie-Hellman (CDH) assumption.

B Trapdoor mercurial commitments

A *trapdoor mercurial commitment scheme* is defined by the following set of algorithms (the definitions given in this section are taken from [9]):

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

$\mathsf{HCom}_{pk}(m)$ Given a message m, this algorithm computes a hard commitment C to m using the public key pk and returns some auxiliary information aux.

$\mathsf{HOpen}_{pk}(m, \mathsf{aux})$ The hard opening algorithm produces a hard decommitment π to the message m correlated to $(C, \mathsf{aux}) = \mathsf{HCom}_{pk}(m)$.

$\mathsf{HVer}_{pk}(m, C, \pi)$ The hard verification algorithm $\mathsf{HVer}_{pk}(m, C, \pi)$ accepts (outputs 1) only if π proves that C is a hard commitment to m.

$\mathsf{SCom}_{pk}()$ produces a soft commitment C and an auxiliary information aux. We observe that a soft commitment string C is created to no message in particular.

$\mathsf{SOpen}_{pk}(m, \mathsf{flag}, \mathsf{aux})$ produces a soft decommitment τ (also known as *"tease"*) to a message m. The parameter $\mathsf{flag} \in \{\mathbb{H}, \mathbb{S}\}$ points out if τ corresponds to a hard commitment $(C, \mathsf{aux}) = \mathsf{HCom}_{pk}(m)$ or to a soft commitment $(C, \mathsf{aux}) = \mathsf{SCom}_{pk}()$. A soft decommitment τ to m says that "if the commitment C produced together with aux can be opened at all, then it would open to m".

$\mathsf{SVer}_{pk}(m, C, \tau)$ checks if τ is a valid decommitment for C to m. If it outputs 1 and τ corresponds to a hard commitment C to m, then C could be hard-opened to m.

$\mathsf{Fake}_{pk,tk}()$ produces a "fake" commitment C which at the beginning is not bound to any message. It also returns an auxiliary information aux.

$\mathsf{HEquiv}_{pk,tk}(m, \mathtt{aux})$ The hard equivocation algorithm generates a hard decommitment π for $(C, \mathtt{aux}) = \mathsf{Fake}_{pk,tk}()$ to the message m. A fake commitment is quite similar to a soft commitment with the additional property that it can be hard-opened.

$\mathsf{SEquiv}_{pk,tk}(m, \mathtt{aux})$ generates a soft decommitment τ to m using the auxiliary information produced by the Fake algorithm.

To satisfy the correctness property we require that $\forall m \in \mathcal{M}$ the following statements are false only with negligible probability:

1. if $(C, \mathtt{aux}) = \mathsf{HCom}_{pk}(m)$:

$$\mathsf{HVer}_{pk}(m, C, \mathsf{HOpen}_{pk}(m, \mathtt{aux})) = 1$$

$$\mathsf{SVer}_{pk}(m, C, \mathsf{SOpen}_{pk}(m, \mathbb{H}, \mathtt{aux})) = 1$$

2. If $(C, \mathtt{aux}) = \mathsf{SCom}_{pk}()$:

$$\mathsf{SVer}_{pk}(m, C, \mathsf{qSOpen}_{pk}(m, \mathbb{S}, \mathtt{aux})) = 1.$$

3. If $(C, \mathtt{aux}) = \mathsf{Fake}_{pk,tk}()$:

$$\mathsf{HVer}_{pk}(m, C, \mathsf{HEquiv}_{pk,tk}(m, \mathtt{aux})) = 1$$

$$\mathsf{SVer}_{pk}(m, C, \mathsf{SEquiv}_{pk,tk}(m, \mathtt{aux})) = 1$$

Security properties We require that a trapdoor mercurial commitment scheme satisfies the following security properties:

- **Mercurial binding**. Having knowledge of pk, it is computationally infeasible for an algorithm $\mathcal{A}$ to come up with C, m, π, m', π' such that either one of the following cases holds:
  - π is a valid hard decommitment for C to m and π' is a valid hard decommitment for C to m', with $m \neq m'$. We call such case a "hard collision".
  - π is a valid hard decommitment for C to m and π' is a valid soft decommitment for C to m', with $m \neq m'$. We call such case a "soft collision".
- **Mercurial hiding**. There exists no PPT adversary $\mathcal{A}$ that, knowing pk, can find a message $m \in \mathcal{M}$ for which it can distinguish $(C, \mathsf{SOpen}_{pk}(m, \mathbb{H}, \mathtt{aux}))$ from $(C', \mathsf{SOpen}_{pk}(m, \mathbb{S}, \mathtt{aux}'))$, where $(C, \mathtt{aux}) = \mathsf{HCom}_{pk}(m)$ and $(C', \mathtt{aux}') = \mathsf{SCom}_{pk}()$.
- **Equivocations**. There exists no PPT adversary $\mathcal{A}$ that, having knowledge of the public key pk and the trapdoor key tk, can win in the following games with non-negligible probability. In such games $\mathcal{A}$ must tell apart the "real" world from its corresponding "ideal" world. The challenger 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.
  - **HHEquivocation.** $\mathcal{A}$ chooses $m \in \mathcal{M}$ and gives it to the challenger. If $b = 0$ the challenger gives to $\mathcal{A}$ a pair (C, π) such that: $(C, \mathtt{aux}) = \mathsf{HCom}_{pk}(m)$ and $\pi = \mathsf{HOpen}_{pk}(m, \mathtt{aux})$. Otherwise it returns (C, π) such that: $(C, \mathtt{aux}) = \mathsf{Fake}_{pk,tk}()$ and $\pi = \mathsf{HEquiv}_{pk,tk}(m, \mathtt{aux})$.
  - **HSEquivocation.** $\mathcal{A}$ chooses $m \in \mathcal{M}$ and gives it to the challenger. If $b = 0$ the challenger gives to $\mathcal{A}$ a real commitment-decommitment tuple (C, τ) such that: $(C, \mathtt{aux}) = \mathsf{HCom}_{pk}(m)$ and $\tau = \mathsf{SOpen}_{pk}(m, \mathbb{H}, \mathtt{aux})$. Otherwise the challenger produces an ideal tuple with: $(C, \mathtt{aux}) = \mathsf{Fake}_{pk,tk}()$ and $\tau = \mathsf{SEquiv}_{pk,tk}(m, \mathtt{aux})$.

- **SSEquivocation.** If $b = 0$ the challenger generates $(C, \mathsf{aux}) = \mathsf{SCom}_{pk}()$ and gives C to $\mathcal{A}$. Then $\mathcal{A}$ chooses $m \in \mathcal{M}$, gives such m to the challenger and receives $\mathsf{SOpen}_{pk}(m, \mathbb{S}, \mathsf{aux})$. Otherwise if $b = 1$ $\mathcal{A}$ first gets $\mathsf{Fake}_{pk,tk}()$, then it chooses $m \in \mathcal{M}$, gives it to the challenger and finally receives $\mathsf{SEquiv}_{pk,tk}(m, \mathsf{aux})$.

At some point $\mathcal{A}$ outputs b' as its guess for b and wins if $b' = b$.
As claimed in [8] it is easy to see that the mercurial hiding is implied by the HHEquivocation and HSEquivocation.

C Requirements for trapdoor q-mercurial commitments

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

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, π, m', π' such that either one of the following cases holds:
  - π is a valid hard decommitment for C to (m, i) and π' is a valid hard decommitment for C to (m', i), with $m \neq m'$. We call such case a "hard collision".
  - π is a valid hard decommitment for C to (m, i) and π' 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 π.
  - **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 [8]) it is easy to see that the q-mercurial hiding is implied by the q-HSEquivocation and q-SSEquivocation.

D Zero-Knowledge Databases

Here we formally describe the properties of a zero-knowledge EDB. Let $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Prove}, \mathsf{V})$ be an EDB system as defined in section 2.2. We say that it is a *zero-knowledge EDB* if:

1. *Completeness.* $\forall$ databases D and $\forall x \in [D]$

$$Pr\begin{bmatrix} CRS & \xleftarrow{\$} & \mathsf{Setup}(1^k); (ZPK, ZSK) & \xleftarrow{\$} \\ \mathsf{Commit}(CRS, D); & & & \\ \pi_x \leftarrow \mathsf{Prove}(CRS, ZSK, x) & : & \mathsf{V}(CRS, ZPK, x, \pi_x) & = \\ D(x) & & & \end{bmatrix} = 1 - \epsilon(k)$$

   where $\epsilon(k)$ is negligible in k.
2. *Soundness.* $\forall x \in \{0,1\}^*$ and $\forall$ PPT algorithms P'

$$Pr\begin{bmatrix} CRS & \xleftarrow{\$} & \mathsf{Setup}(1^k); (ZPK', \pi_x, \pi'_x) & \xleftarrow{\$} \\ P'(CRS, D): & & & \\ \mathsf{V}(CRS, ZPK', x, \pi_x), \mathsf{V}(CRS, ZPK', x, \pi'_x) & & & \neq \\ \bot \wedge & & & \\ \mathsf{V}(CRS, ZPK', x, \pi_x) \neq \mathsf{V}(CRS, ZPK', x, \pi'_x) & & & \end{bmatrix}$$

   is negligible.
3. *Zero-Knowledge.* Let us assume that exists an oracle that, queried on a key x, returns the correct value associated to x in a database D if $x \in [D]$, otherwise it returns $\bot$. Then there exists a simulator $SIM = (\mathsf{SimSetup}, \mathsf{SimCom}, \mathsf{SimProve})$, where $\mathsf{SimProve}$ has oracle access to D, such that $\forall$ adversaries $\mathcal{A}$, $\forall k \in \mathbb{N}$ and $\forall$ databases D, $\mathcal{A}$'s view $View(k)$ in the real game is indistinguishable from its view $View'(k)$ in the game played by SIM.

$$View(k) = \left\{ \begin{array}{c} CRS \xleftarrow{\$} \mathsf{Setup}(1^k); (ZPK, ZSK) \xleftarrow{\$} \mathsf{Commit}(CRS, D); \\ (x_1, s_1) \xleftarrow{\$} \mathcal{A}(CRS, ZPK); \pi_{x_1} \leftarrow \mathsf{Prove}(ZSK, x_1): \\ (x_2, s_2) \xleftarrow{\$} \mathcal{A}(CRS, ZPK, s_1, \pi_{x_1}); \\ \pi_{x_2} \leftarrow \mathsf{Prove}(x_2, ZSK); \\ \vdots \\ : ZPK, x_1, \pi_{x_1}, x_2, \pi_{x_2} ... \end{array} \right\}$$

$$View'(k) = \left\{ \begin{array}{c} (CRS', t) \xleftarrow{\$} \mathsf{SimSetup}(1^k); (ZPK', ZSK') \xleftarrow{\$} \mathsf{SimCom}(); \\ (x_1, s_1) \xleftarrow{\$} \mathcal{A}(CRS, ZPK'); \pi'_{x_1} \leftarrow \mathsf{SimProve}^D(x_1, ZSK') : \\ (x_2, s_2) \xleftarrow{\$} \mathcal{A}(CRS', ZPK', s_1, \pi'_{x_1}); \\ \pi'_{x_2} \leftarrow \mathsf{SimProve}^D(x_2, ZSK'); \\ \vdots \\ : ZPK', x_1, \pi'_{x_1}, x_2, \pi'_{x_2} \end{array} \right\}$$

D.1 Zero-Knowledge Databases from trapdoor q-mercurial commitments

In this section we recall the construction given by Catalano *et al.* in [9] that builds ZK EDBs from trapdoor q-mercurial commitments, trapdoor mercurial commitments and collision resistant hash functions.

Let T_h be the complete q-ary tree of height h, with q^h leaves. Let $\mathcal{U}_k$ be a universe of size 2^k and T_h be the complete q-ary tree of height h. Each element $x \in \mathcal{U}$ can be associated to a leaf of T_h and all the nodes in the path from the root to x can be labeled with the q-ary encoding of x. Thus ϵ (the empty string) is the label for the root. If v is a non-leaf node, then $v1, \cdots, vq$ are the labels for its q sons. Let $S \subseteq \mathcal{U}_k$ and consider the following two subsets of T_h: $TREE(S)$ and $FRONTIER(S)$. $TREE(S)$ is the subtree of T_h containing all the nodes in the paths from the leaves in S to the root. $FRONTIER(S) = \{v : v \notin TREE(S) \wedge parent(v) \in TREE(S)\}$.

The construction is given by describing the following algorithms:

$\mathsf{Setup}(1^k)$ Output the common reference string $CRS = (PK, PKM, H)$ where PK is the public key of a trapdoor q-mercurial commitment scheme $\mathcal{QC}$, PKM the public key of a trapdoor mercurial commitment scheme $\mathcal{C}$ and H is a collision resistant hash function.

$\mathsf{Commit}(CRS, D)$

1. Let $S = \{H(x) : x \in D, D(x) \neq \perp\}$. Construct the tree $T = TREE(S) \bigcup FRONTIER(S)$.
2. $\forall$ leaf-nodes $H(x)$ set:
$$n_{H(x)} = \begin{cases} H(y) & \text{if } D(x) = y, \\ 0 & \text{if } D(x) = \perp. \end{cases}$$
compute a hard commitment $(C_{H(x)}, \mathsf{aux}_{H(x)}) = \mathsf{HCom}_{PKM}(n_{H(x)})$ and set $m_{H(x)} = H(C_{H(x)})$.
3. $\forall$ internal nodes u such that $u \in FRONTIER(S) : (C_u, \mathsf{aux}_u) = \mathsf{qSCom}_{PK}()$.
4. $\forall$ internal nodes $u \in TREE(S)$: $(C_u, \mathsf{aux}_u) = \mathsf{qHCom}_{PK}(m_{u1}, \cdots, m_{uq})$, store aux_u in u and set $m_u = H(C_u)$ if $u \neq \epsilon$.
5. Output the commitment of the root: $ZPK = (C_\epsilon)$. The secret key ZSK contains all the stored elements.

$\mathsf{Prove}(CRS, ZSK, x)$ The prover algorithm produces a proof π_x of the database value of x. We distinguish between two cases:

1. $\mathbf{D}(\mathbf{x}) = \mathbf{y}$. The proof π_x contains the commitments and the hard openings in the path from the leaf-node $H(x)$ toward the root: $\pi_x = \Big\{ y, C_{H(x)}, HO_{H(x)} = \mathsf{HOpen}_{PKM}(H(y), \mathsf{aux}_{H(x)}), \{C_u, qHO_u = \mathsf{qHOpen}_{PK}(m_v, i, \mathsf{aux}_u)\}_{v=H(x),\cdots,\epsilon-1} \Big\}$ with $u = parent(v), i = index_u(v)$.

2. $\mathbf{D}(\mathbf{x}) = \perp$. The prover first checks if the leaf node $H(x)$ is in the tree T. If $H(x) \notin T$, let $w \in FRONTIER(S)$ be the root of the missing subtree of T_k containing $H(x)$. The prover builds the subtree rooted in w using the same algorithm as in Commit. The proof π_x contains the commitments and the soft openings in the path from $H(x)$ toward the root ϵ:
$\Big\{C_{H(x)}, SO_{H(x)} = \mathsf{SOpen}_{PKM}(0, \mathbb{S}, \mathsf{aux}_{H(x)}),$
$\{C_u, qSO_u = \mathsf{qSOpen}_{PK}(m_v, i, \mathbb{H}/\mathbb{S}, \mathsf{aux}_u)\}_{v=H(x),\cdots,\epsilon-1}\Big\}$
with $u = parent(v), i = index_u(v)$.

$\mathsf{V}(CRS, ZPK, x, \pi_x)$ We consider two cases according to the type of π_x:

1. $D(x) = y$.
   (a) Check if $\mathsf{HVer}_{PKM}(H(y), C_{H(x)}, HO_{H(x)}) = 1$.
   (b) Compute $m_{H(x)} = H(C_{H(x)})$.
   (c) Let $v = parent(H(x))$ and i such that $H(x) = vi$. Check if $\mathsf{qHVer}_{PK}(m_{H(x)}, i, C_v, qHO_v) = 1$.
   (d) compute $m_v = H(C_v)$ and iterate as above for $u = parent(v), \cdots, \epsilon - 1$.
   (e) Check if $\mathsf{qHVer}_{PK}(m_u, i, ZPK, qHO_\epsilon) = 1$, where $u = i$ is the first node in the path from the root toward $H(x)$.
   (f) If none of the tests above fails, output y, otherwise output $\perp$.
2. $D(x) = \perp$.
   (a) Check if $\mathsf{SVer}_{PKM}(0, C_{H(x)}, SO_{H(x)}) = 1$
   (b) Compute $m_{H(x)} = H(C_{H(x)})$.
   (c) Let $v = parent(H(x))$ and i such that $H(x) = vi$. Check if $\mathsf{qSVer}_{PK}(m_{H(x)}, i, C_v, qSO_v) = 1$.
   (d) Compute $m_v = H(C_v)$ and and iterate as above for $u = parent(v), \cdots, \epsilon - 1$.
   (e) Check if $\mathsf{qSVer}_{PK}(m_u, i, ZPK, qSO_\epsilon) = 1$, where $u = i$ is the first node in the path from the root toward $H(x)$.
   (f) If none of the tests above fails output *out*, else output $\perp$.

E Proofs

E.1 Proof of theorem 1

We prove the theorem by showing that all the properties of trapdoor q-mercurial commitments are satisfied.

q-Mercurial Binding. Here we show that neither kind of collisions (i.e. hard or soft ones) are possible. We start by discussing the case of hard collisions. The case of soft collisions is basically the same.

Hard Collisions. Assume that there is an adversary $\mathcal{A}$ that manages to succeed in the following experiment with non negligible probability. $\mathcal{A}$ receives in input the public key of the qTMC he intends to attack and produces a commitment C together with two valid (but different!) openings (τ_i, τ_i'). In particular assume that, wlog, both τ_i and τ_i' are valid decommitments for messages in position i. For simplicity assume that $\tau_i = (\Lambda_i, C_i, \pi_i)$ opens to m_i at position i, while $\tau_i' = (\Lambda_i', C_i', \pi_i')$ opens to m_i' at position i (and of course $m_i \neq m_i'$). We distinguish two (mutually exclusive) types of openings the adversary might produce:

Type I $C_i = C_i'$

Type II $C_i \neq C'_i$

Type I. If the adversary produces two different openings of type I it is easy to see that one can break the mercurial binding of TMC. Indeed both τ_i and τ'_i are correctly verified (i.e. $\mathsf{HVer}_{PK_{\mathsf{TMC}}}(C_i, m_i, \pi_i) = 1$ and $\mathsf{HVer}_{PK_{\mathsf{TMC}}}(C_i, m'_i, \pi'_i) = 1$) and thus the tuple $(C_i, m_i, \pi_i, m'_i, \pi'_i)$ is a hard collision for the mercurial binding of TMC.

Type II. If the adversary produces two openings of type II then it is easy to see that this case breaks the position binding property of the vector commitment VC. Indeed, by definition, both Λ_i and Λ'_i are correctly verified openings (i.e. $\mathsf{VC.Ver}_{vpk}(C, C_i, i, \Lambda_i) = 1$ and $\mathsf{VC.Ver}_{vpk}(C, C'_i, i, \Lambda'_i) = 1$). Thus one can output $(C, C_i, C'_i, i, \Lambda_i, \Lambda'_i)$ to break the position binding.

Soft Collisions. This proof is almost identical to that given for hard collision and is omitted. The only difference is that if the adversary outputs two openings of type I, here we break the mercurial binding of TMC by finding a soft collision for it.

Equivocation and Hiding. In all the cases we show that it is infeasible for an adversary, on input both the public key and the trapdoor, to distinguish between a real commitment/decommitment tuple and a fake/equivocation one.

In general $\mathcal{A}$ receives in input the pair (pk, tk), gives an index i and a tuple $(m_1, \ldots, m_q)$ and receives back a tuple (S_b, τ_b) that is generated using different algorithms according to a secret bit b. If we consider $\tau_b = (\Lambda_i, C_i, \pi_i)$ and observe the pair (S_b, Λ_i), it is easy to see that in all the cases these elements have always the same distribution no matter of which algorithm has generated them. Thus the only elements that might have different distributions according to b are C_i and π_i, but these are indistinguishable provided that TMC satisfies hiding and equivocations. □

E.2 Proof of theorem 3

We prove the theorem by showing that the scheme satisfies the position binding property. More precisely, assume for sake of contradiction that there exists an efficient adversary that produces two valid openings to two different messages at the same position, then we show how to build an algorithm $\mathcal{B}$ that breaks either the collision resistance of the hash function family or the RSA assumption.

$\mathcal{B}$ receives on input (N, z, e, H), where e is an $(\ell + 1)$-bit prime, and uses $\mathcal{A}$ to produce either a collision on H or a value y such that $z = y^e \bmod N$. $\mathcal{B}$ embeds the RSA challenge as follows. It randomly chooses $j \in \{1, \ldots, q\}$ and sets $e_j = e$ and $a = z$. The rest of the public key is computed as required by the $\mathsf{VC.KeyGen}$ algorithm.

$\mathcal{B}$ runs $\mathcal{A}(vpk)$ and gets back $(S, m_i, m'_i, i, \Lambda_i, \Lambda'_i)$ where $m_i \neq m'_i$ and both Λ_i and Λ'_i are correctly verified. If $i \neq j$ $\mathcal{B}$ aborts the simulation. Otherwise it proceeds as follows. It computes $c_i \leftarrow H(m_i)$ and $c'_i \leftarrow H(m'_i)$. If $c_i = c'_i$ $\mathcal{B}$ halts and outputs the collision (c_i, c'_i) for H. Otherwise from the equations $S = S_i^{c_i} \Lambda_i^{e_i}$, $S = S_i^{c'_i} \Lambda'^{e_i}_i$ we get

$$S_i^{c_i - c'_i} = (\Lambda'_i / \Lambda_i)^{e_i}$$

Let $c = c_i - c'_i$ and $\Lambda = \Lambda'_i / \Lambda_i$, the equation above can be rewritten as

$$a^{c \prod_{j \neq i} e_j} = (\Lambda)^{e_i}$$

If $\Lambda = 1$ then we can factor with non-negligible probability (this is because, being a a random element in $\mathbb{Z}_N$ it is of high order with very high probability, see [13] for details). Thus assuming $\Lambda \neq 1$ we can apply Shamir's trick [24] to get an e_i-th root of a. In particular, since $\gcd(c\prod_{j\neq i} e_j, e_i) = 1$, by the extended Euclidean Algorithm we can compute two integers λ, μ such that $\lambda c\prod_{j\neq i} e_j + \mu e_i = 1$. This leads to the equation

$$a = \Lambda^{\lambda e_i} a^{\mu e_i}$$

thus $\Lambda^{\lambda} a^{\mu}$ is the required root.

E.3 Proof of theorem 5

We prove the theorem by showing that the scheme satisfies the position binding property. Assume for sake of contradiction that there exists an efficient adversary that produces two valid openings to two different messages at the same position, then we show how to build an algorithm $\mathcal{B}$ that breaks the Inversion Computational Diffie-Hellman assumption (Inv-CDH).

$\mathcal{B}$ receives as input a tuple (g, g^a) and its goal is to compute $g^{1/a}$.

First, the simulator picks a random $i \xleftarrow{\$} [q]$ as a guess for the index i on which $\mathcal{A}$ will break the position binding. Next, $\mathcal{B}$ takes $z_j \xleftarrow{\$} \mathbb{Z}_p$, $\forall j \in [q], j \neq i$, and computes:

$$g_j = g^{z_j},\ h_j = (g^a)^{\prod_{k=1,k\neq i,j}^{q} z_k}\ \forall j \in [q] \setminus \{i\},\ \ g_i = g^a, h_i = g^{\prod_{k=1,k\neq i}^{q} z_k},$$
$$\forall r, j \in [q], r \neq j \wedge r, j \neq i : h_{r,j} = (g^a)^{\prod_{k=1,k\neq i,j,r}^{q} z_k},\ \ h_{i,j} = g^{\prod_{k=1,k\neq i,j}^{q} z_k}\ \ h_{r,i} = g^{\prod_{k=1,k\neq i,r}^{q} z_k}$$

$\mathcal{B}$ sets $vpk = (g, \{g_i, h_i\}_{i\in[q]}, \{h_{i,j}\}_{i,j\in[q],i\neq j})$. It runs $\mathcal{A}$ on input vpk, and it is supposed to get back $(C, m_j, m'_j, j, \Lambda_j, \Lambda'_j)$ such that: $m_j \neq m'_j$ and both Λ_j and Λ'_j are correctly verified.

If $i \neq j$, then $\mathcal{B}$ aborts the simulation. Otherwise it returns

$$g^{1/a} = (\Lambda_i/\Lambda'_i)^{1/((m_i - m'_i)(\prod_{k\neq i} z_k))}.$$

To see that the output is correct, observe that since the two openings verify correctly, then it holds:

$$e(C, g) = e(h_i^{m_i}, g)e(\Lambda_i, g_i) = e(h_i^{m'_i}, g)e(\Lambda'_i, g_i)$$

which means that

$$e(h_i^{m_i - m'_i}, g) = e(\Lambda_i/\Lambda'_i, g^a)$$

From this we have:

$$(\Lambda_i/\Lambda'_i)^a = h_i^{(m_i - m'_i)} = g^{(\prod_{k\neq i} z_k)(m_i - m'_i)}$$

which justifies the correctness of $\mathcal{B}$'s output.

F A trapdoor mercurial commitment from the RSA assumption

In this section we recall the mercurial commitment scheme originally proposed by Gennaro and Micali in [12].

$\mathsf{KeyGen}(k, \ell)$ First, randomly choose two $k/2$-bit primes p_1, p_2 and set $N = p_1 p_2$. Then select a random $(\ell + 1)$-bit prime e, such that $\gcd(e, \phi(N)) = 1$ and a random element T in $\mathbb{Z}_N$. The public key $\mathrm{PK}_{\mathsf{TMC}}$ is set as (N, T, e) while the trapdoor tk is the e-th root of T. We assume that messages are strings of ℓ bits (otherwise one can use a collision-resistant hash function to shrink the size of messages to ℓ bits).

$\mathsf{HCom}_{PK_{\mathsf{TMC}}}(m)$ Choose two random elements $R, r \in \mathbb{Z}_N$ and set $s = Tr^e \bmod N$ and $C = s^m R^e$. Output the commitment (C, s) and the auxiliary information $\mathsf{aux}_{\mathsf{TMC}} = (m, R, r)$.

$\mathsf{HOpen}_{PK_{\mathsf{TMC}}}(m, \mathsf{aux}_{\mathsf{TMC}})$ Parse $\mathsf{aux}_{\mathsf{TMC}}$ as (m', R, r). Output $\perp$ if $m \neq m'$ and (R, r) otherwise.

$\mathsf{HVer}_{PK_{\mathsf{TMC}}}(C, s, m, R, r)$ The hard verification algorithm returns 1 if and only if $s = Tr^e \bmod N$ and $C = s^m R^e$.

$\mathsf{SCom}_{PK_{\mathsf{TMC}}}()$ Choose two random elements $R, r \in \mathbb{Z}_N$ and set $s = r^e \bmod N$ and $C = R^e$. Output the soft commitment (C, s) and the auxiliary information $\mathsf{aux}_{\mathsf{TMC}} = (R, r)$.

$\mathsf{SOpen}_{PK_{\mathsf{TMC}}}(m, \mathsf{flag}, \mathsf{aux}_{\mathsf{TMC}})$. If $\mathsf{flag} = \mathbb{H}$ proceed as follows. Parse $\mathsf{aux}_{\mathsf{TMC}}$ as (m', R, r), if $m' \neq m$ return $\perp$, otherwise output R. If $\mathsf{flag} = \mathbb{S}$, then output $R' = r^{-m} R \bmod N$.

$\mathsf{SVer}_{PK_{\mathsf{TMC}}}(C, s, m, R)$ The soft verification algorithm returns 1 if and only if $C = s^m R^e$.

$\mathsf{Fake}_{PK_{\mathsf{TMC}}, tk}()$ This is the same as the SCom algorithm.

$\mathsf{HEquiv}_{PK_{\mathsf{TMC}}, tk}(m, \mathsf{aux}_{\mathsf{TMC}})$ Parse $\mathsf{aux}_{\mathsf{TMC}}$ as (R, r) and output (R', r') where $r' = r \cdot (tk)^{-1} \bmod N$ and $R' = r^{-m} R \bmod N$.

$\mathsf{SEquiv}_{PK_{\mathsf{TMC}}, tk}(m, \mathsf{aux}_{\mathsf{TMC}})$ Parse $\mathsf{aux}_{\mathsf{TMC}}$ as (R, r) and output $R' = r^{-m} R \bmod N$.

G The MRK trapdoor mercurial commitment scheme

Here we briefly describe the trapdoor mercurial commitment scheme based on the Discrete Logarithm Assumption given by Micali *et al.* in [18], and later formalized by Chase *et al.* in [10].

Let G be a group of prime order p in which the Discrete Logarithm is hard. The scheme is defined by the following algorithms:

- $\mathsf{KeyGen}(1^k)$: the key generation algorithm selects a random generator $g \in G$, picks a random integer $x \in \mathbb{Z}_p^*$ and sets $h = g^x$. The public key is $pk = (g, h)$ while the trapdoor is $tk = x$;
- $\mathsf{HCom}_{pk}(m)$: selects two random integers $r, s \in \mathbb{Z}_p^*$ and then sets $h_r = h^r, C = g^m h_r^s$. The hard commitment to m is the pair (C, h_r), while r, s represent the auxiliary information;
- $\mathsf{HOpen}_{pk}(m, \mathsf{aux})$: produces a hard decommitment π which contains the auxiliary information $\mathsf{aux} = r, s$;
- $\mathsf{HVer}_{pk}((C, h_r), m, \pi)$: uses the public key to check if $C = g^m h_r^s$ and $h_r = h^r$. It returns 1 if both the tests are valid, 0 otherwise;
- $\mathsf{SCom}_{pk}()$: produces a soft commitment (C, h_r) where $h_r = g^r$ and $C = h_r^s$, for r, s randomly chosen in $\mathbb{Z}_p^*$;
- $\mathsf{SOpen}_{pk}(m, \mathsf{aux})$: produces a soft decommitment $\tau = s'$ to the message m for a commitment produces using aux. It computes $s' = s - m/r$;
- $\mathsf{SVer}_{pk}((C, h_r), m, \tau)$: the soft verification algorithm checks if $C = g^m h_r^{s'}$;
- $\mathsf{Fake}_{pk,tk}()$: does the same computation as SCom;
- $\mathsf{HEquiv}_{pk,tk}(m, \mathsf{aux})$: uses the trapdoor key to create a valid hard decommitment $\pi = r, s'$ to m where $s' = s - m/rx$;
- $\mathsf{SEquiv}_{pk,tk}(m, \mathsf{aux})$: returns a soft decommitment $\tau = s'$, with s' computed as in the hard equivocation.

H Updatable mercurial commitments

First, we provide a formal definition of updatable mercurial commitments since in [17] it is only given as a sketch.

An *updatable (trapdoor) mercurial commitment* (uTMC for brevity) is defined like a mercurial commitment with the following additional algorithms:

$\mathsf{Update}_{pk}(C, \mathtt{aux}, m')$ This algorithm is run by the committer who produced C (and holds $\mathtt{aux}$). It takes as input a message m' and outputs a new commitment C' and an update information U. Regardless of the type of C, the updated commitment C' is always a hard commitment.

$\mathsf{ProofUpdate}_{pk}(C, \pi, m', U)$ This algorithm is run by any user who holds a proof π for a commitment C. Given the update information U and the new message m', $\mathsf{ProofUpdate}$ outputs the updated commitment C' and an updated proof π' which will be valid w.r.t. C' and m'. The updated proof π' will be of the same type of π.

The mercurial binding is defined as usual, namely for any PPT adversary it is computationally infeasible to open a commitment (even an updated one) to two different messages. Then we require updatable mercurial commitments to satisfy the usual mercurial hiding and equivocations properties of TMCs, but we also require such properties to hold even for updated commitments, namely when the adversary can see the update information[10].

Remark 7 (On the security of updatable mercurial commitments). Given the hiding and equivocations properties depicted above, one may wonder if they are the strongest possible properties one can have in a scenario where updates are issued. In fact we notice they are not. For example, there is no guarantee that an updated commitment is hiding w.r.t. the original message contained before the update. However, as already noticed by Liskov in [17], such weakness is not a problem for the means of building updatable ZK EDBs as in this case the committed messages are commitments themselves.

However since updatable mercurial commitments might have applications outside the context of updatable ZK EDBs, it would be interesting to consider more general security properties. In Appendix K we address this problem and we provide augmented security properties for updatable mercurial commitments that we call *Updatable Hiding*, *Updatable mercurial hiding* and *Updatable Equivocations*. The schemes given by Liskov in [17] (both the generic and the one based on Discrete Log) and the scheme we propose in the following section *do not* satisfy the new security definition. Therefore we will show in Appendix K how to realize an updatable mercurial commitment from RSA that is secure under updatable hiding and equivocations.

H.1 An updatable mercurial commitment from RSA

Here we extend the scheme of Gennaro and Micali [12] which is recalled in appendix F to make it updatable via the following additional algorithms:

$\mathsf{Update}_{pk}(C, s, \mathtt{aux}_{\mathsf{TMC}}, m')$ If C is a hard commitment, parse $\mathtt{aux}$ as (m, R, r) and proceed as follows. Pick a random $r' \xleftarrow{\$} \mathbb{Z}_N$ and compute: $s' = r'^e T \bmod N$ and $C' = s'^{m'} R^e \bmod N$. Otherwise, if C is a soft commitment (i.e. $\mathtt{aux} = (R, r)$) that has been teased to a message m then proceed as follows. Pick a random $r' \xleftarrow{\$} \mathbb{Z}_N$, set $R' = r^{-m} R$ and compute $s' = r'^e T \bmod N$ and $C' = s'^{m'} R'^e \bmod N$. Finally output the updated commitment (C', s') and the update information $U = r'$.

$\mathsf{ProofUpdate}_{pk}(C, \pi, m', U)$ Who holds a proof π that was valid for C, can use the update information U to compute the updated commitment C' to m' and produce a new proof π' which will be valid w.r.t. C' and m'. The updated commitment is $(C' = s'^{m'} R^e \bmod N, s' = r'^e T \bmod N)$.

[10] Notice that since an updated commitment is always a hard commitment, in this case we are interested only to Hiding and HHEquivocation.

If π is a soft opening (i.e. $\pi = R$) the updated proof π' remains the same. Otherwise, if π is a hard opening (i.e. $\pi = (R, r)$) the updated proof is $\pi' = (R, r')$.

Theorem 8. *If the RSA assumption holds, then the scheme given above is an updatable trapdoor mercurial commitment.*

It is easy to see that the same security proof for the scheme of Gennaro and Micali [12] still holds in this case.

I Efficient Independent Zero-Knowledge Databases from Strong RSA

Gennaro and Micali introduced in [12] the notion of *Independent Zero-Knowledge Databases*. Roughly speaking the *independence* property prevents an adversary from correlating its database and proofs to those of an honest prover. In particular Gennaro and Micali show that previous constructions of ZK EDBs do not satisfy this stronger notion of security and proposed new ones achieving it.

The definition of strong-independence considers a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. $\mathcal{A}_1$ sees ℓ database commitments $Com_1, \ldots, Com_\ell$ generated by honest provers and is allowed to make queries on the underlying databases $Db_1, \ldots, Db_\ell$. Finally $\mathcal{A}_1$ must output a new database commitment Com. Then two copies of the second-stage adversary $\mathcal{A}_2$ are run: the first is given oracle access to provers that output proofs of elements in Db_i; the second copy of $\mathcal{A}_2$ has access to an oracle that makes proofs for databases Db'_i such that, for all $i = 1$ to ℓ, Db_i and Db'_i agree on the elements queried by $\mathcal{A}_1$.

More formally, we say that a ZK EDB system is *strongly-independent* if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and databases $Db_1, \ldots, Db_\ell, Db'_1, \ldots, Db'_\ell$ there exists a simulator $SIM = (\mathsf{SimSetup}, \mathsf{SimCom}, \mathsf{SimProve})$ such that the following probability

$$Pr\left[\begin{array}{l}(CRS, t) \leftarrow \mathsf{SimSetup}(1^k);\\ (Com_i, st_i) \leftarrow \mathsf{SimCom}(CRS, t) \forall i = 1, \ldots, \ell;\\ (Com, st) \leftarrow \mathcal{A}_1^{\mathsf{SimProve}^{Db_i}(st_i, Com_i)}(CRS, st);\\ (x, \pi_x) \leftarrow \mathcal{A}_2^{\mathsf{SimProve}^{Db_i}(st_i, Com_i)}(CRS, st);\\ (x, \pi'_x) \leftarrow \mathcal{A}_2^{\mathsf{SimProve}^{Db'_i \vdash_{Q_i} Db_i}(st_i, Com_i)}(CRS, st) :\\ \mathsf{V}(CRS, Com, x, \pi_x) \neq \mathsf{V}(CRS, ZPK, x, \pi'_x) \neq\\ \perp \wedge ((Com \neq Com_i \forall i) \vee (\exists i : Com = Com_i \wedge x \notin\\ Q'_i))\end{array}\right]$$

is negligible in k.

In the definition above, Q_i is the list of queries made by $\mathcal{A}_1$ to the oracle $\mathsf{SimProve}^{Db_i}(st_i, Com_i)$ and we denote with $Db'_i \vdash_{Q_i} Db_i$ that the database Db'_i is restricted to agree with Db_i on the elements of Q_i.

Gennaro and Micali [12] give also two weaker notions of independence. The first one (*weak independence*) considers an adversary $\mathcal{A}_1$ that outputs the commitment without making any query. The second one (simply *independence*) is the same as strong-independence except that $\mathcal{A}_2$ is not allowed to copy one of the honest provers' commitments.

I.1 Strongly-Independent Zero-Knowledge Databases from concise (multi-trapdoor) q-mercurial commitments

The construction of strongly-independent ZK EDBs of Gennaro and Micali [12] uses as building block a *multi-trapdoor mercurial commitment* (which we define below) that is combined with a signature scheme and a collision resistant hash function. Libert and Yung [16] later adapted this construction to work in the q-ary tree setting of Catalano *et al.* [9] by using, as building block, a *concise multi-trapdoor q-mercurial commitment*.

Before describing this construction, we first give the definition of *multi-trapdoor concise q-mercurial commitments*. Such a scheme is a collection of the following algorithms:

$\mathsf{qKeyGen}(1^k, q)$ is a probabilistic algorithm that takes in input a security parameter k, the parameter q indicating the length of valid sequences and outputs a pair of public/private keys (pk, tk). The public key also contains the description of a tag space $\mathcal{T}$.

$\mathsf{qHCom}_{pk}(m_1, \cdots, m_q, tag)$ Given an ordered tuple of q messages and $tag \in \mathcal{T}$, qHCom computes a hard commitment C to $(m_1, \cdots, m_q)$ under (pk, tag) and returns an auxiliary information aux.

$\mathsf{qHOpen}_{pk}(m, i, tag, \mathsf{aux})$ Let $(C, \mathsf{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q, tag')$, if $m = m_i$ and $tag = tag'$ the hard opening algorithm produces a hard decommitment π. Otherwise it returns an error message.

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

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

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

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

$\mathsf{qTrapGen}_{pk,tk}(tag)$ given $tag \in \mathcal{T}$, it outputs a tag-specific trapdoor key tk_{tag}.

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

$\mathsf{qHEquiv}_{pk,tk}(m, i, tag, \mathsf{aux})$ The hard equivocation algorithm generates a hard decommitment π to the message of m at position i for $(C, \mathsf{aux}) = \mathsf{qFake}_{pk,tk_{tag}}(tag)$. A q-fake commitment is very similar to a soft commitment with the additional property that it can be hard-opened.

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

As in the previous definitions, such a commitment is said *concise* if the size of the hard and soft openings is independent of q and if an opening to a message m_i does not reveal information about the other m_j's for $j \neq i$.

The correctness requirements are essentially the same as those for trapdoor q-mercurial commitment schemes with the addition of the tag.

The security properties are an extension of the q-mercurial binding, hiding and equivocations that are given in section 2.1.

q-Mercurial binding The adversary is allowed to choose a set of strings $tag_1, \ldots, tag_n \in \mathcal{T}$ that are given to a Challenger who replays back with pk and $tk_{tag_1}, \ldots, tk_{tag_n}$ (that are generated using $\mathsf{qTrapGen}_{pk,tk}(tag_i)$). Then the q-mercurial binding holds if it is computationally infeasible for an algorithm $\mathcal{A}$ to come up with $tag^* \in \mathcal{T} \setminus \{tag_1, \ldots, tag_n\}$, a commitment C and two different openings m, i, π, m', π' such that $\mathsf{qHVer}_{pk}(C, m, i, tag^*, \pi) = 1$ and $\mathsf{qHVer}_{pk}(C, m', i, tag^*, \pi') = 1$, or $\mathsf{qHVer}_{pk}(C, m, i, tag^*, \pi) = 1$ and $\mathsf{qSVer}_{pk}(C, m', i, tag^*, \pi') = 1$.

q-Mercurial hiding and Equivocations These properties are basically the same of standard q-mercurial commitments. The only difference is that here the adversary is allowed to specify in each game a tag under which the challenge commitment is created. This means that the tag is not necessarily hidden in the commitment, but this suffices for the means of our application.

We notice that our definition is slightly different from that given in [16] in that we require the qSCom and qFake algorithms to take in input the tag. Though this definition is weaker, it is sufficient for the purposes of our application, namely constructing independent ZK EDBs.

Given a multi-trapdoor concise q-mercurial commitment, a signature scheme $\Sigma = (\mathsf{kg}, \mathsf{sig}, \mathsf{ver})$ and a collision resistant hash function $H : \{0,1\}^* \to \mathcal{T}$ (where $\mathcal{T}$ is the tag space), strongly-independent ZK EDBs can be realized as follows. To commit to a database one first generates a signing key pair $(vk, sk) \xleftarrow{\$} \mathsf{kg}()$ and then builds a ZK EDB using the construction of [9] (which is recalled in section D.1) where all the (multi-trapdoor) commitments are generated with the same tag $H(vk)$. If Com is the database commitment obtained from the procedure above, then the committer outputs (Com, vk). To generate a proof for an element x, the prover produces π_x, as usual, and outputs (π_x, σ_x) where $\sigma_x = \mathsf{sig}_{sk}(Com, x)$. Finally, the verification procedure checks the validity of both π_x (w.r.t. the tag $H(vk)$) and the signature σ_x.

The strong-independence property of this construction is implied by the q-mercurial binding of the multi-trapdoor qTMC, the collision resistance of H and the security (unforgeability under chosen-message attack) of Σ. The proof is the same as that given in [16]. As already noticed above, our definition of multi-trapdoor q-mercurial commitments is slightly different, but we observe that this does not invalidate the existing proof in [16] as the tag $H(vk)$ is known by the simulator when it computes fake commitments.

I.2 Building multi-trapdoor concise q-mercurial commitments

It is quite easy to see that a multi-trapdoor concise q-mercurial commitment can be built using the construction of section 3.1 where the standard mercurial commitment is replaced with a multi-trapdoor one. This means that the tag is employed only to compute the underlying mercurial commitments.

Multi-trapdoor mercurial commitments were introduced by Gennaro and Micali in [12], though they did not give an explicit definition for them. The primitive is defined essentially in the same way as multi-trapdoor q-mercurial commitments when restricted to the case $q = 1$.

Below we recall the scheme of Gennaro and Micali [12] which is based on Strong RSA.

$\mathsf{KeyGen}(k, \ell)$ First, randomly choose two $k/2$-bit primes p_1, p_2 and set $N = p_1 p_2$. Next, let $H : \{0,1\}^* \leftarrow \{0,1\}^{\ell+1}$ be a collision resistant hash function that outputs prime numbers[11] (chosen at random from a collision resistant function family $\mathcal{H}$), $\mathcal{T}$ be the description of a tag space and T a random element in $\mathbb{Z}_N$. The public key $\mathrm{PK}_{\mathsf{TMC}}$ is set as $(N, T, H, \mathcal{T})$ while the trapdoor tk is the factorization (p_1, p_2) of N. We assume that messages are strings of ℓ bits (otherwise one can use a collision resistant hash function to shrink the size of messages to ℓ bits).

$\mathsf{HCom}_{PK_{\mathsf{TMC}}}(m, tag)$ Choose two random elements $R, r \in \mathbb{Z}_N$ and set $s = Tr^{H(tag)} \bmod N$ and $C = s^m R^{H(tag)}$. Output the commitment (C, s) and the auxiliary information $\mathsf{aux}_{\mathsf{TMC}} = (m, tag, R, r)$.

$\mathsf{HOpen}_{PK_{\mathsf{TMC}}}(m, tag, \mathsf{aux}_{\mathsf{TMC}})$ Parse $\mathsf{aux}_{\mathsf{TMC}}$ as (m', tag', R, r). Output $\perp$ if $(m, tag) \neq (m', tag')$ and $\pi = (R, r)$ otherwise.

$\mathsf{HVer}_{PK_{\mathsf{TMC}}}(C, s, m, tag, \pi)$ The hard verification algorithm returns 1 if and only if $s = Tr^{H(tag)} \bmod N$ and $C = s^m R^{H(tag)}$.

$\mathsf{SCom}_{PK_{\mathsf{TMC}}}(tag)$ Choose two random elements $R, r \in \mathbb{Z}_N$ and set $s = r^{H(tag)} \bmod N$ and $C = R^{H(tag)}$. Output the soft commitment (C, s) and the auxiliary information $\mathsf{aux}_{\mathsf{TMC}} = (tag, R, r)$.

$\mathsf{SOpen}_{PK_{\mathsf{TMC}}}(m, tag, C, s, \mathsf{flag}, \mathsf{aux}_{\mathsf{TMC}})$. If $\mathsf{flag} = \mathbb{H}$ proceed as follows. Parse $\mathsf{aux}_{\mathsf{TMC}}$ as (m', tag', R, r). If $(m', tag') \neq (m, tag)$ return $\perp$, otherwise output $\pi = R$. If $\mathsf{flag} = \mathbb{S}$ then output $\pi = R' = r^{-m} R \bmod N$.

$\mathsf{SVer}_{PK_{\mathsf{TMC}}}(m, tag, C, \pi)$ The soft verification algorithm returns 1 if and only if $C = s^m R^{H(tag)}$.

$\mathsf{TrapGen}_{PK_{\mathsf{TMC}}, tk}(tag)$ Use the factorization of N to compute $tk_{tag} = T^{1/H(tag)}$ and output tk_{tag}.

$\mathsf{Fake}_{PK_{\mathsf{TMC}}, tk_{tag}}()$ This is the same as the SCom algorithm.

$\mathsf{HEquiv}_{PK_{\mathsf{TMC}}, tk_{tag}}(m, tag, \mathsf{aux}_{\mathsf{TMC}})$ Parse $\mathsf{aux}_{\mathsf{TMC}}$ as (tag, R, r) and output (R', r') where $r' = r \cdot (tk_{tag})^{-1} \bmod N$ and $R' = r^{-m} R \bmod N$.

$\mathsf{SEquiv}_{PK_{\mathsf{TMC}}, tk}(m, tag, \mathsf{aux}_{\mathsf{TMC}})$ Parse $\mathsf{aux}_{\mathsf{TMC}}$ as (tag, R, r) and output $R' = r^{-m} R \bmod N$.

Theorem 9. *The scheme above is a multi-trapdoor mercurial commitment under the Strong-RSA assumption.*

We defer the interested reader to [12] for the proof of this theorem.

I.3 Strongly-Independent Zero-Knowledge Databases with short proofs from Strong RSA

When we instantiate the construction of section I.1 with the concise multi-trapdoor qTMC sketched in the previous section[12], we get a strongly-independent ZK EDB with security based on Strong RSA and whose proofs have size linear in the height h of the q-ary tree. The size of the proofs is the same as in the "non-independent" case, namely $6h-1$ and $5h-1$ elements of $\mathbb{Z}_N$ for each proof of membership and non-membership respectively.

J Additional constructions of Concise Vector Commitments: a variant with constant size public key

A limitation in the above construction is that the size of the public key is linear in q. If we care about the application to ZKS this may be a limitation as it affects the size of the common

[11] See [11] for the details on how to implement such a function.

[12] We mean the scheme obtained by combining our vector commitment based on RSA and the multi-trapdoor mercurial commitment based on Strong RSA.

reference string and thus q cannot be increased too much. Here we show how to slightly change the construction given in the previous section to obtain a variant where the public key has constant size size. Unfortunately the new scheme is computationally less efficient, but it is interesting as it leads to the first construction of a qTMC where even the public key is of size independent of q.

In order to obtain such variant we apply a technique used by Waters and Hohenberger in [14] to obtain a deterministic function that outputs primes.

We describe below the modified key generation algorithm.

$\mathsf{VC.KeyGen}(k, \ell, q)$ Randomly choose two $k/2$-bit primes p_1, p_2 and set $N = p_1 p_2$. Then randomly select a seed s for a pseudorandom function $f_s : \{0,1\}^* \rightarrow \{0,1\}^{\ell+1}$ and a binary string $c \xleftarrow{\$} \{0,1\}^{\ell+1}$ and define the function $F_{s,c}(j) = f_s(i,j) \oplus c$ where $i \geq 1$ is the smallest index such that $f_s(i,j) \oplus c$ is odd and prime. Next, let $H : \{0,1\}^* \leftarrow \{0,1\}^{\ell}$ be a collision resistant hash function (chosen at random from a family $\mathcal{H}$ of collision resistant functions) and a be a random element in $\mathbb{Z}_N$. For $i = 1$ to q we implicitly define

$$S_i = a^{\prod_{j=1, j\neq i}^{q} e_j}$$

where $e_j = F_{s,c}(j)\ \forall j = 1, \ldots, q$. The public key vpk is set as (N, a, s, c, H).

The rest of the scheme is basically the same where the elements S_i and e_i are reconstructed using the function $F_{s,c}(\cdot)$.

The scheme above can be proved secure using the same proof of Theorem 3 extended with the techniques presented in [14] (Theorem 4.1) to show that the function $f_s(\cdot)$ is "good enough" to have a correct simulation. The reader is referred to [14] for details omitted here. This allows us to state the following theorem

Theorem 10. *If the RSA assumption holds, H is collision resistant and f is a pseudorandom function, then the scheme given above is a concise vector commitment.*

K Updatable mercurial commitments with updatable hiding

As already noticed in section H, it is interesting to consider security properties for updatable mercurial commitments that are reasonable in general, not only for the application of building ZK EDBs. In this section we address this problem and provide augmented security properties for hiding and equivocations of updatable mercurial commitments. Finally we will also propose a scheme that realizes our stronger security definition.

Roughly speaking we would like that the usual hiding and equivocations properties should be preserved even when updates are issued. For this reason, we define the following augmented properties:

Updatable Hiding We say that an uTMC has *updatable hiding* if any PPT adversary $\mathcal{A}$ has at most negligible probability of winning the following game.
The adversary is allowed to choose two messages m_0, m_1 and then receives C which is a commitment to m_b for a randomly chosen $b \xleftarrow{\$} \{0,1\}$. Then $\mathcal{A}$ is allowed to choose other two messages m'_0, m'_1 and gets (C', U) that are an updated commitment and the update information for m'_d where $d \xleftarrow{\$} \{0,1\}$ is chosen at random. Finally $\mathcal{A}$ outputs two bits b', d' and we say that it *wins* if $b' = b$ and $d' = d$.

Informally this property says that given C, C' and U it is still infeasible to extract information about the committed messages. The definition above is intended for *computational* updatable hiding. The corresponding perfect and statistical definitions easily follow by making no assumptions on the power of the adversary.

Updatable mercurial hiding and equivocations We slightly modify the games of mercurial hiding and all equivocations (HH, HS, SS) and we say that an uTMC satisfies these properties if any PPT adversary can win in such games with at most negligible probability. Each game is modified in a way similar to updatable hiding. Namely, after having seen a commitment/decommitment tuple, we add a final phase where the adversary is allowed to see an updated commitment C' and the related update information U for a message m' of its choice. Then we require that the type of C remains hidden even when the adversary is given C, U. Since an updated commitment is always a hard commitment (by its definition) we are not interested into concealing its type.

It is easy to see that if a scheme satisfies these augmented properties, then it also satisfies the standard mercurial hiding and equivocations. Moreover if the mercurial commitment is proper, then it suffices to check only HH and SS equivocations as these imply all the other properties.

Remark 11. While in standard mercurial commitments the mercurial hiding implies the standard one, this no longer holds here. Indeed, observe that the games of updatable hiding and updatable mercurial hiding differ in that in the latter one does not concern about hiding the type of C'. This means that updatable hiding has to be independently checked.

K.1 Updatable mercurial commitments with updatable hiding from RSA

In this section we show an updatable mercurial commitment scheme that satisfies our security definitions given above under the RSA assumption. The scheme is obtained by extending the one given in section 6.

$\mathsf{KeyGen}(k, \ell)$ First, randomly choose two $k/2$-bit primes p_1, p_2 and a prime $e > N$.Next, let T a random element in $\mathbb{Z}_N$ and ck be the public key of a standard commitment $\mathcal{C} = (\mathsf{Com}, \mathsf{Open}, \mathsf{Ver})$. The public key $\mathrm{PK}_{\mathsf{TMC}}$ is set as (N, T, e, ck) while the trapdoor tk is the e-th root of T.

$\mathsf{HCom}_{PK_{\mathsf{TMC}}}(m)$ Choose random elements $R, r, \alpha \in \mathbb{Z}_N$. Then compute $s = Tr^e \bmod N$, $C = s^{(m+\alpha)}R^e$, $(\mu, \mathsf{aux}_\mu) = \mathsf{Com}_{ck}(\alpha)$. Finally output the commitment (C, s, μ) and the auxiliary information $\mathsf{aux}_{\mathsf{TMC}} = (m, R, r, \alpha, \mathsf{aux}_\mu)$.

$\mathsf{HOpen}_{PK_{\mathsf{TMC}}}(m, \mathsf{aux}_{\mathsf{TMC}})$ Parse $\mathsf{aux}_{\mathsf{TMC}}$ as $(m', R, r, \alpha, \mathsf{aux}_\mu)$. Output $\perp$ if $m \neq m'$ and (R, r, α, π_μ) otherwise (where $\pi_\mu \leftarrow \mathsf{Open}_{ck}(\alpha, \mathsf{aux}_\mu)$).

$\mathsf{HVer}_{PK_{\mathsf{TMC}}}(C, s, \mu, m, R, r, \alpha, \pi_\mu)$ The hard verification algorithm returns 1 if and only if $s = Tr^e \bmod N$, $C = s^{(m+\alpha)}R^e$ and $\mathsf{Ver}_{ck}(\mu, \alpha, \pi_\mu) = 1$.

$\mathsf{SCom}_{PK_{\mathsf{TMC}}}()$ Choose random elements $R, r, \alpha \in \mathbb{Z}_N$. Then compute $s = r^e \bmod N$, $C = R^e$ and $(\mu, \mathsf{aux}_\mu) = \mathsf{Com}_{ck}(\alpha)$. Output the soft commitment (C, s, μ) and the auxiliary information $\mathsf{aux}_{\mathsf{TMC}} = (R, r, \alpha, \mathsf{aux}_\mu)$.

$\mathsf{SOpen}_{PK_{\mathsf{TMC}}}(m, \mathsf{flag}, \mathsf{aux}_{\mathsf{TMC}})$. If $\mathsf{flag} = \mathbb{H}$ proceed as follows. Parse $\mathsf{aux}_{\mathsf{TMC}}$ as $(m', R, r, \alpha, \mathsf{aux}_\mu)$, if $m' \neq m$ return $\perp$, otherwise output R, α, π_μ where $\pi_\mu \leftarrow \mathsf{Open}_{ck}(\alpha, \mathsf{aux}_\mu)$.

If $\mathsf{flag} = \mathbb{S}$, then output (R', α, π_μ) such that $R' = r^{-(m+\alpha)}R \bmod N$, and $\pi_\mu \leftarrow \mathsf{Open}_{ck}(\alpha, \mathsf{aux}_\mu)$.

$\mathsf{SVer}_{PK_{\mathsf{TMC}}}(C, s, \mu, R, \alpha, \pi_\mu)$ The soft verification algorithm returns 1 if and only if $C = s^{(m+\alpha)}R^e$ and $\mathsf{Ver}_{ck}(\mu, \alpha, \pi_\mu) = 1$.

$\mathsf{Fake}_{PK_{\mathsf{TMC}},tk}()$ This is the same as the SCom algorithm.

$\mathsf{HEquiv}_{PK_{\mathsf{TMC}},tk}(m, \mathtt{aux}_{\mathsf{TMC}})$ Parse $\mathtt{aux}_{\mathsf{TMC}}$ as $(R, r, \alpha, \mathtt{aux}_\mu)$ and output $(R', r', \alpha, \pi_\mu)$ where $r' = r \cdot (tk)^{-1} \bmod N$, $R' = r^{-(m+\alpha)}R \bmod N$ and $\pi_\mu \leftarrow \mathsf{Open}_{ck}(\alpha, \mathtt{aux}_\mu)$.

$\mathsf{SEquiv}_{PK_{\mathsf{TMC}},tk}(m, \mathtt{aux}_{\mathsf{TMC}})$ Parse $\mathtt{aux}_{\mathsf{TMC}}$ as $(R, r, \alpha, \mathtt{aux}_\mu)$ and output (R', α, π_μ) where $R' = r^{-(m+\alpha)}R \bmod N$ and $\pi_\mu \leftarrow \mathsf{Open}_{ck}(\alpha, \mathtt{aux}_\mu)$.

$\mathsf{Update}_{pk}(C, s, \mu, \mathtt{aux}_{\mathsf{TMC}}, m')$ If C is a hard commitment, parse $\mathtt{aux}_{\mathsf{TMC}}$ as $(m, R, r, \alpha, \mathtt{aux}_\mu)$ and proceed as follows. Pick a random $r' \xleftarrow{\$} \mathbb{Z}_N$ and compute: $s' = r'^e T \bmod N$ and $C' = s'^{(m'+\alpha)}R^e \bmod N$. Otherwise, if C is a soft commitment (i.e. $\mathtt{aux}_{\mathsf{TMC}} = (R, r, \alpha, \mathtt{aux}_\mu)$) that has been teased to a message m then proceed as follows. Pick a random $r' \xleftarrow{\$} \mathbb{Z}_N$, set $R' = r^{-(m+\alpha)}R$ and compute $s' = r'^e T \bmod N$ and $C' = s'^{(m'+\alpha)}R'^e \bmod N$. Finally output the updated commitment (C', s', μ) and the update information $U = r'$.

$\mathsf{ProofUpdate}_{pk}(C, m', \pi, U)$ Who holds a proof π that was valid for C, can use the update information U to compute the updated commitment C' to m' and produce a new proof π' which will be valid w.r.t. C' and m'. The updated commitment is $(C' = s'^{(m'+\alpha)}R^e \bmod N, s' = r'^e T \bmod N, \mu)$. If π is a soft opening (i.e. $\pi = (R, \alpha, \pi_\mu)$) the updated proof π' remains the same. Otherwise, if π is a hard opening (i.e. $\pi = (R, r, \alpha, \pi_\mu)$) the updated proof is $\pi' = (R, r', \alpha, \pi_\mu)$.

Theorem 12. *If the RSA assumption holds and $\mathcal{C}$ is a commitment, then the scheme given above is an updatable trapdoor mercurial commitment.*

Proof. We prove the theorem by showing that each property is satisfied. The proof of the mercurial binding is basically the same as that of the scheme of Gennaro and Micali [12]. Let (C, s, μ) be a commitment and let (R, r, α, π_μ) and $(R', r', \alpha', \pi'_\mu)$ be two hard openings to m and m' respectively (such that $m \neq m'$). Here we can have two different cases. If $\alpha \neq \alpha'$ then we can break the binding of $\mathcal{C}$. Otherwise, if $\alpha = \alpha'$ we can break RSA as in the proof of [12] (with the additional case that $m = m' \bmod \phi(N)$ in which one can factor N).

The scheme clearly satisfies the usual hiding and equivocations properties for the same argument of [12]. Finally we show below that it satisfies our stronger properties of updatable hiding and equivocations.

Updatable Hiding and equivocations. In order to prove that the scheme satisfies updatable hiding we recall below the view of the adversary in this game:

$$\mathcal{D}_{b,d} = \{C = s^{(m_b+\alpha)}R^e, s = r^e T, \mu = \mathsf{Com}_{ck}(\alpha), C' = s'^{(m'_d+\alpha)}R^e, s' = r'^e T, U = r'\}.$$

One can observe that for any choice of the messages (m_b, m'_d) there exist α, R that satisfy the equations above. This means that, without μ, the distributions $\mathcal{D}_{b,d}$ for $b, d \in \{0, 1\}$ are perfectly indistinguishable for any messages m_0, m_1, m'_0, m'_1. Therefore, if the commitment μ is hiding, updatable hiding also holds.

A similar argument applies in the case of the equivocations properties.

In the case of updatable HHEquivocation the adversary sees either one of the following tuples:

$$\{C = s^{(m+\alpha)}R^e, s = r^e T, \mu = \mathsf{Com}(\alpha), R, r, \pi_\mu, T^{1/e}, C' = s'^{(m'+\alpha)}R^e, s' = r'^e T, U = r'\}$$

$$\{C = R^e, s = r^e, \mu = \mathsf{Com}(\alpha), (r^{-(m+\alpha)}R), (rT^{1/e}), \pi_\mu, T^{1/e}, C' = s'^{(m'+\alpha)}(r^{-(m+\alpha)}R)^e, s' = r'^e T, U = r'\}$$

where it chooses m and m'. The two views are clearly perfectly indistinguishable.

Finally, in the case of updatable SSEquivocation it is trivial to see that the two views are exactly the same:

$$\{C = R^e, s = r^e, \mu = \mathsf{Com}(\alpha), (r^{-(m+\alpha)}, R), \pi_\mu, T^{1/e}, C' = s'^{(m'+\alpha)}(r^{-(m+\alpha)}R)^e, s' = r'^e T, U = r'\}$$

$$\{C = R^e, s = r^e, \mu = \mathsf{Com}(\alpha), (r^{-(m+\alpha)}, R), \pi_\mu, T^{1/e}, C' = s'^{(m'+\alpha)}(r^{-(m+\alpha)}R)^e, s' = r'^e T, U = r'\}$$