Zero-Knowledge Sets with Short Proofs

Dario Catalano*, Mario Di Raimondo*, Dario Fiore†, Mariagrazia Messina‡
* Dipartimento di Matematica ed Informatica – Università di Catania, Italy
{catalano,diraimondo}@dmi.unict.it
† École Normale Supèrieure – CNRS-INRIA, Paris, France
dario.fiore@ens.fr
‡ Microsoft Italia
mariame@microsoft.com

***Abstract*—Zero Knowledge Sets, introduced by Micali, Rabin and Kilian in 2003, allow a prover to commit to a secret set S in a way such that it can later prove, non interactively, statements of the form $x \in S$ (or $x \notin S$), without revealing any further information (on top of what explicitly revealed by the inclusion/exclusion statements above) on S, not even its size. Later, Chase *et al.* abstracted away the Micali, Rabin and Kilian's construction by introducing an elegant new variant of commitments that they called (trapdoor) mercurial commitments. Using this primitive, it was shown how to construct zero knowledge sets from a variety of assumptions (both general and number theoretic).**

This paper introduces the notion of trapdoor q-mercurial commitments (qTMCs), a notion of mercurial commitment that allows the sender to commit to an *ordered* sequence of exactly q messages, rather than to a single one. Following the previous work it is shown how to construct ZKS from qTMCs and collision resistant hash functions.

Then, it is presented an efficient realization of qTMCs that is secure under the so called Strong Diffie Hellman assumption, a number theoretic conjecture recently introduced by Boneh and Boyen. Using such scheme as basic building block, it is obtained a construction of ZKS that allows for proofs that are much shorter with respect to the best previously known implementations. In particular, for an appropriate choice of the parameters, our proofs are up to 33% shorter for the case of proofs of membership, and up to 73% shorter for the case of proofs of non membership. Experimental tests confirm practical time performances.

***Index Terms*—Security, integrity and protection, Public Key Cryptosystems.**

I. Introduction

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

An extended abstract of this paper appears in the proceedings of EUROCRYPT 2008 [5].

Sponsored by MIUR under project "*Priv-Ware*" (contract n. 2007JXH7ET).

The third and fourth authors entirely did the present work while students at University of Catania.

The notion of Zero Knowledge Sets (ZKS) was recently introduced by Micali, Rabin and Kilian [20] to address exactly this problem. Informally, ZKS allow a prover P to commit to an arbitrary (but finite) set S in a way such that P can later prove statements of the form $x \in S$ or $x \notin S$ without revealing any significant information about S (not even its size!). As already pointed out in [20], the notion of Zero-Knowledge Sets can be easily extended to encompass the more general notion of elementary databases (EDB). In a nutshell, an elementary database is a set S with the additional property that each $x \in S$ comes with an associated value $S(x)$. In the following we will refer to ZKS to include Zero-Knowledge EDB as well.

The solution by Micali *et al.* is non interactive and works in the so called *shared random string* model (i.e. where a random string, built by some trusted third party, is made available to all participants) building upon a very clever utilization of a simple commitment scheme, originally proposed by Pedersen [26].

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

Since Pedersen's commitment relies on the intractability of the discrete logarithm, so does the construction in [20]. Later, Chase *et al.* [6] abstracted away Micali *et al.*'s solution and described the exact properties a commitment scheme should possess in order to allow a similar construction. This led to an elegant new variant of commitments, that they called *mercurial commitment*.

Informally, a mercurial commitment is a commitment scheme where the binding requirement is somewhat relaxed in order to allow for two decommitment procedures: an *hard* and a *soft* one. At committing time, the sender can decide as whether to create an *hard* commitment or a *soft* one, from the message m he has in mind. Hard commitments are like standard ones, in the sense that they can be (hard or soft)

opened only with respect to the message that was originally used to construct the commitment. Soft commitments, on the other hand, allow for more freedom, as they cannot be hard opened in any way, but they can be soft opened to any arbitrary message. An important requirement of mercurial commitments is that, hard and soft commitments should look alike to any polynomially bounded observer.

Using this new primitive, Chase *et al.* proved that it is possible to construct ZKS from a variety of assumptions (number theoretic or general)[1]. Their most general implementation, shows that (non interactive) ZKS can be constructed, in the shared random string model, assuming non interactive zero knowledge proofs (NIZK) [2] and collision resistant hash functions [9][2]. Moreover, they showed that collision resistant hash functions are necessary to construct ZKS, as they are implied by the latter. Finally, Catalano, Dodis and Visconti [4] gave a construction of (trapdoor) mercurial commitments from one way functions in the shared random string model. This result completed the picture as it showed that collision resistant hash functions are necessary and sufficient for non interactive ZKS in the shared random string model.

OUR CONTRIBUTION. All the constructions above, build upon the common idea of constructing an authenticated Merkle tree of depth k where each internal node is a mercurial commitment (rather than the hash) of its two children. Very informally, to prove that a given $x \in \{0,1\}^k$ belongs to the committed set S, the prover simply opens all the commitments in the path from the root to the leaf labeled by x (more details about this methodology will be given later on). Thus the length of the resulting proof is $k \cdot d$, where d denotes the length of the opening of the mercurial commitment, and k has to be chosen so that 2^k is larger than the size of any "reasonably" large set S[3]. Assuming $k = 128$ and $d = O(k)$, as it is the case for all known implementations, this often leads to very long proofs.

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

In this paper, we present a new construction of ZKS that allows for much shorter proofs, with respect to the best currently known implementation (which is the Micali *et al.* construction when implemented on certain classes of elliptic curves. From now on we will use the acronym MRK to refer to such an implementation). As shown in Section V, the proposed construction keep a practical interest with a good time efficiency as proved by some experimental tests.

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

Following [6], [20], we show how to construct ZKS from qTMCs and collision resistant hash functions. This step is rather simple but very useful for our goal, as it reduces the task of realizing efficient ZKS to the task of realizing efficient qTMCs. Indeed, even though the proposed transformation allows us to use a "flat" Merkle tree (i.e. with branching factor q, rather than two), it *does not* lead, by itself, to shorter proofs.

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

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

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

In addition, we provide a more detailed comparison between the MRK scheme and the ours by giving an implementation of both schemes. We ran experimental tests to measure their performances in terms of time efficiency and the results of such experiments showed that our proposal is widely practical in time efficiency and it is even better in some operations when

[1]More precisely, they require the mercurial commitment to be *trapdoor* as well. Very informally, this means that the scheme comes with a trapdoor information *tk* (normally not available to anyone) that allows to completely destroy the binding property of the commitment

[2]It is known that one can construct NIZK under the assumption that trapdoor permutations exist or under the assumption that verifiable random functions (VRF) exist [14], [21]. These two assumptions, however, are, as far as we currently know, incomparable.

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

compared with MRK.

ZERO KNOWLEDGE SETS VS SIGNATURES. The idea of obtaining short proofs by changing the authentication procedure to deal with a "flat" authentication tree, is reminiscent of a technique originally suggested by Dwork and Naor [10], in the context of digital signature schemes. In a nutshell, the Dwork-Naor method allows to increase the branching factor of the tree without inflating the signature size. This is achieved, by, basically, authenticating each node with respect to its parent, but without providing its siblings.

Adapting this idea to work to the case of zero knowledge sets, presents several non trivial technical difficulties[4]. The main problem comes from the fact that, in ZKS, one has to make sure that a dishonest prover cannot construct two, both valid, proofs for the statements $x \in S$ and $x \notin S$. Such a requirement imposes limitations just not present when dealing with digital signatures.

OTHER RELATED WORK. Very recently Libert and Yung [15] solved the problem left open in [5] of having qTMCs where hard openings are of size independent of q. The construction proposed in [15] is based on the q-DHE assumption and allows to build ZKS where even membership proofs can be made shorter by increasing the branching factor of the tree.

Ostrovsky, Rackoff and Smith [24] described a construction that allows a prover to commit to a database and to provide answers that are consistent with the commitment. Their solution can handle more elaborate queries than just membership ones. Moreover they also consider the issue of adding privacy to their protocol. However their construction requires interaction (at least if one wants to avoid the use of random oracles) and requires the prover to keep a counter for the questions asked so far.

Gennaro and Micali introduced in [12] the notion of independent zero knowledge sets. Informally, independent ZKS protocols prevent an adversary from successfully correlate her set to the one of a honest prover. Their notion of independence also implies that the resulting ZKS protocol is non-malleable and requires a new commitment scheme that is both independent and mercurial. We do not consider such an extension here.

Liskov [18] considered the problem of updating zero-knowledge databases. In [18] definitions for updatable zero knowledge databases are given, together with a construction based on verifiable random functions [21] and mercurial commitments. The construction, however, is in the random oracle model [1].

Prabhakaran and Xue [27] 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.

Finally it may be of interest to observe that the accumulator proposed by Nguyen in [22] has a construction similar to our q-mercurial commitment scheme.

[4]It is probably instructive to mention the fact that, indeed, the Dwork Naor solution, and its improvements such as [8], *do not* work in our setting

ROAD MAP. The paper is organized as follows. In section II we introduce the notion of trapdoor q mercurial commitments and provide the relevant definitions for zero knowledge sets. Section III is devoted to the construction of ZKS from trapdoor q mercurial commitments. In Section IV we show how to construct efficient qTMCs from the Strong Diffie Hellman Assumption. A formal and experimental efficiency analysis is given in Section V. Finally conclusions and directions for future work are given in Section VI.

II. PRELIMINARIES

Before presenting our results we briefly recall some basic definitions. In what follows we will denote with k a security parameter. Denote with $\mathbb{N}$ the set of natural numbers and with $\mathbb{R}^+$ the set of positive real numbers. We say that a function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible if and only if for every positive polynomial $P(k)$ there exists a $k_0 \in \mathbb{N}$ such that for all $k > k_0$ $\epsilon(k) < 1/P(k)$. If S is a set, we write $x \xleftarrow{\$} S$ to indicate the process of selecting x uniformly at random in S. Let A be a probabilistic algorithm. We denote with $x \xleftarrow{\$} A$ the process of running A and assigning the its output to x.

A. Trapdoor q-mercurial commitments

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

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

- qKeyGen$(1^k, q)$: is a probabilistic algorithm that takes in input a security parameter k and the number q of committed values and outputs a pair of public/private keys

(pk, tk). This algorithm is usually run by a trusted party that makes the public key available to all parties that want to use the scheme.

- $\mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$: given an ordered tuple of messages, qHCom computes a hard commitment C to $(m_1, \cdots, m_q)$ using the public key pk and returns some auxiliary value $\mathtt{aux}$ containing all the information used to generate C.
- $\mathsf{qHOpen}_{pk}(m, i, \mathtt{aux})$: let $(C, \mathtt{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$, if $m = m_i$ the hard opening algorithm $\mathsf{qHOpen}_{pk}(m, i, \mathtt{aux})$ produces a hard decommitment π. The algorithm returns an error message otherwise.
- $\mathsf{qHVer}_{pk}(m, i, C, \pi)$: the hard verification algorithm accepts (outputs 1) only if π proves that C is created to a tuple $(m_1, \cdots, m_q)$ such that $m_i = m$.
- $\mathsf{qSCom}_{pk}()$: produces a soft commitment C and an auxiliary information $\mathtt{aux}$. A soft commitment string C is created to no specific sequence of messages.
- $\mathsf{qSOpen}_{pk}(m, i, \mathsf{flag}, \mathtt{aux})$: 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 $(C, \mathtt{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$ or to a soft commitment $(C, \mathtt{aux}) = \mathsf{qSCom}_{pk}()$. The algorithm returns an error message if C is an hard commitment and $m \neq m_i$.
- $\mathsf{qSVer}_{pk}(m, i, C, \tau)$: checks if 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 $\mathtt{aux}$.
- $\mathsf{qHEquiv}_{pk,tk}(m_1, \cdots, m_q, i, \mathtt{aux})$: the non-adaptive hard equivocation algorithm generates a hard decommitment π for $(C, \mathtt{aux}) = \mathsf{qFake}_{pk,tk}()$ 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.
- $\mathsf{qSEquiv}_{pk,tk}(m, i, \mathtt{aux})$: the soft equivocation algorithm generates a soft decommitment τ to m of position i using the auxiliary information produced by the qFake algorithm. We notice that it does not need to have the entire commitment sequence to be specified in input.

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

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

1) 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 a PPT 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 $(m_1, \cdots, m_q) \in \mathcal{M}^q$ for which it can distinguish $(C, \{\mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathtt{aux})\}_{i=1,\ldots,q})$ from $(C', \{\mathsf{qSOpen}_{pk}(m_i, i, \mathbb{S}, \mathtt{aux}')\}_{i=1,\ldots,q})$, where $(C, \mathtt{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$ and $(C', \mathtt{aux}') = \mathsf{qSCom}_{pk}()$.
- **Equivocations**. In the following games $\mathcal{A}$ should not 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.
  - **q-HEquivocation.** The adversary $\mathcal{A}$ chooses a sequence $(m_1, \cdots, m_q) \in \mathcal{M}^q$ and gives it to the challenger. If $b = 0$ the challenger computes $(C, \mathtt{aux}) = \mathsf{qHCom}_{pk}(m_1, \cdots, m_q)$, $\pi_i = \mathsf{qHOpen}_{pk}(m_i, i, \mathtt{aux})$ and $\tau_i = \mathsf{qSOpen}_{pk}(m_i, i, \mathbb{H}, \mathtt{aux})$ $\forall i = 1, \ldots, q$. Otherwise, if $b = 1$, it computes: $(C, \mathtt{aux}) = \mathsf{qFake}_{pk,tk}()$, $\pi_i = \mathsf{qHEquiv}_{pk,tk}(m_1, \cdots, m_q, i, \mathtt{aux})$ and $\tau_i = \mathsf{qSEquiv}_{pk,tk}(m_i, i, \mathtt{aux})$ $\forall i = 1, \ldots, q$. Then the the adversary is given the commitment C and the openings $\pi_1, \tau_1, \ldots, \pi_q, \tau_q$ for all the chosen messages.
  - **q-SEquivocation.** In this game the challenger picks a random bits $b \xleftarrow{\$} \{0, 1\}$ and proceeds as follows. If $b = 0$ it generates $(C, \mathtt{aux}) = \mathsf{qSCom}_{pk}()$ and gives C to $\mathcal{A}$. Next, $\mathcal{A}$ chooses q messages $(m_1, \ldots, m_q) \in \mathcal{M}^q$, gives them to the challenger and receives back $\mathsf{qSOpen}_{pk}(m, i, \mathbb{S}, \mathtt{aux})$ for all $i = 1, \ldots, q$. If $b = 1$, $\mathcal{A}$ first gets $\mathsf{qFake}_{pk,tk}()$, then it chooses $(m_1, \ldots, m_q) \in \mathcal{M}^q$, gives them to the challenger and gets back $\mathsf{qSEquiv}_{pk,tk}(m, i, \mathtt{aux})$ for all $i = 1, \ldots, q$.

At the end of each game $\mathcal{A}$ outputs a bit b' as its guess for b and we define its advantage as $Adv^{X-equiv}(\mathcal{A}) = 2 \cdot Pr[b' = b] - 1$ (for X=H, S respectively). Then we say that such properties hold is no PPT adversary $\mathcal{A}$ has non-negligible advantage in any of the above games.

As for the case of trapdoor mercurial commitments (see [4]) it is easy to see that the q-mercurial hiding is implied by the q-HEquivocation and q-SEquivocation. Moreover if a scheme is proper in the sense of [4], then the q-HEquivocation property can be simplified by giving to the adversary only hard openings since the soft ones are given implicitly. Indeed, we recall that a mercurial commitment scheme is said *proper* if the soft opening is a proper subset of the hard opening.

RELATION TO THE DEFINITION IN [5]. In this work we slightly change the definition of equivocations with respect to the one given in [5]. The first modification is in that here the adversary of the hiding and equivocations games is allowed to see the openings for all the q indices. Indeed we noticed that the previous definition (where the adversary is given the opening for only one index) is not sufficient for proving the zero-knowledge property in the generic construction of ZKS from qTMCs.

Then we propose a new q-HEquivocation property instead of the previous q-HHEquivocation and q-HSEquivocation. The point is that the previous definition allows to build ZKS only if the scheme is proper.[5] More precisely, from a theoretical point of view, it does not change much as any non-proper scheme can be converted into a proper one by defining the hard opening as the hard opening plus the soft opening. Though our construction of section IV is proper, we prefer to have a statement valid for all kinds of schemes, also because the generic conversion sketched above, may not be efficient. In contrast our new definition allows to build ZKS directly without any requirement on the properness of the qTMC scheme.

Finally we remark that the same observation can be made in the case of standard mercurial commitments. We propose new equivocations definitions also for them and we discuss the relationship with the previous definition in Appendix A.

B. Generic construction of trapdoor q-mercurial commitments

In this section we show that trapdoor q-mercurial commitments exist under the very basic assumption that one-way functions exist. Indeed, from the result of Catalano *et al.* [4] we know that mercurial commitments exist assuming the existence of one-way functions in the common reference string model.

Therefore a trivial solution is to define a trapdoor q-mercurial commitment as an ordered sequence of q mercurial commitments. In order to commit to a sequence of messages $m_1, \ldots, m_q$ the committer creates $\forall i = 1, \ldots, q$ C_i as the commitment to message m_i using the mercurial commitment scheme and then defines $C = (C_1, \ldots, C_q)$. Later, when it wants to open C to m at position j, it opens C_j to m. It is easy to see that this q-mercurial commitment scheme is secure if so is the underlying scheme. Therefore, putting together this scheme and the result of Catalano *et al.* [4] we have that trapdoor q-mercurial commitments can be built assuming only the existence of one-way functions in the common reference string model.

It is clear that the solution given above is not very efficient since the size of the commitment string is linear in q. A more space-efficient generic construction can be instantiated assuming the existence of collision resistant hash functions. To commit to a sequence of messages use the mercurial commitment scheme to create q commitments $C_1, \ldots, C_q$ and then construct a Merkle tree upon the q commitments as leaves. In particular observe that the path from the root until each leaf uniquely defines its position in the sequence. The commitment C will be the root of the tree. Later, to open C to m at position j, open C_j to m and then "open" the path from C_j until the root. In this case the commitment string is constant and independent from q, while the opening cost is $O(\log q)$.

C. Zero-Knowledge Sets

Zero-Knowledge sets [20] allow one to commit to some secret set S and then to, non interactively, produce proofs of the form $x \in S$ or $x \notin S$. This is done without revealing any further information (i.e. that cannot be deduced by the statements above) about S, not even its size. Following the approach of [20], here we focus on the more general notion of zero-knowledge elementary databases (EDB), since the notion of Zero-Knowledge Sets is a special case of zero-knowledge EDBs (see [20] for more details about this). Let $[D]$ be the set of keys associated to a database D. We assume that $[D]$ is a proper subset of $\{0,1\}^*$. If $x \in [D]$, we denote with $y = D(x)$ its associated value in the database D. If $x \notin [D]$ we let $D(x) = \emptyset$. An EDB system is formally defined by the following tuple of algorithms $(\mathsf{Setup}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$:

- $\mathsf{Setup}(1^k)$ takes in input the security parameter k and generates the common reference string CRS together with some trapdoor information tk. This algorithm is run by some trusted party that makes CRS available to all parties that will be going to use the EDB system.
- $\mathsf{P}_1(CRS, D)$, the *committer* algorithm, takes in input the common reference string CRS, a database D and outputs a public key ZPK and a secret key ZSK.
- $\mathsf{P}_2(CRS, ZPK, ZSK, x)$ On input the common reference string CRS, the public key ZPK, the secret key ZSK (which implicitly contains a description of D) and an element x, the *prover* algorithm produces a proof π_x of either $D(x) = y$ or $D(x) = \emptyset$.
- $\mathsf{V}(CRS, ZPK, x, \pi_x)$ The *verifier* algorithm outputs y if $D(x) = y$, $\emptyset$ if $D(x) = \emptyset$ or $\perp$ if the proof π_x is not valid.

We say that $(\mathsf{Setup}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ is a *zero-knowledge EDB* if it satisfies the following properties:

[5]While such statement may be unclear at this stage, it will become more clever when looking at the proof in Section III-B.

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

$$Pr\begin{bmatrix}(CRS,tk) \xleftarrow{\$} \mathsf{Setup}(1^k);\\ (ZPK,ZSK) \xleftarrow{\$} \mathsf{P}_1(CRS,D);\\ \pi_x \leftarrow \mathsf{P}_2(CRS,ZPK,ZSK,x):\\ \mathsf{V}(CRS,ZPK,x,\pi_x) = D(x)\end{bmatrix} = 1-\epsilon(k)$$

where $\epsilon(k)$ is negligible in k. Informally, this means that if $D(x) = y$ then the prover is able to convince a verifier of this fact with overwhelming probability.

2) *Soundness.* A dishonest prover should not be able to prove false statements even if it provides a maliciously chosen public key. More formally, $\forall x \in \{0,1\}^*$ and $\forall$ efficient algorithms P'

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

is negligible in k.

3) *Zero-Knowledge.* This property means that a verifier learns only the values $D(x)$ during its interaction with the prover. This is formally modeled by saying that there exists a simulator $\mathsf{SIM} = (\mathsf{SimSetup}, \mathsf{SimCom}, \mathsf{SimProve})$ such that $\forall k \in N$, for all PPT adversaries $\mathcal{A}$ and for all efficiently computable[6] (adversarially chosen) databases D, the views of the adversary $\mathcal{A}$ in the following games are indistinguishable.

- **Game Real.** In this game a common reference string CRS is generated according to the Setup algorithm and the security parameter and $\mathcal{A}$ is allowed to choose a database D and give it to the prover. Then the adversary receives a public key ZPK from the prover (computed using P_1). Finally it adaptively chooses a sequence of database keys $x_1,\ldots,x_n$ for which it wants to see proofs for the correct values in D: $D(x_1),\ldots,D(x_n)$. Thus $\mathcal{A}$'s view in this game is:

$$View(k) = \left\{\begin{array}{c}(CRS,tk) \xleftarrow{\$} \mathsf{Setup}(1^k);\\ (D,st) \xleftarrow{\$} \mathcal{A}(CRS,tk);\\ (ZPK,ZSK) \xleftarrow{\$} \mathsf{P}_1(CRS,D);\\ (x_1,st_1) \xleftarrow{\$} \mathcal{A}(st,ZPK);\\ \pi_{x_1} \leftarrow \mathsf{P}_2(CRS,ZSK,x_1):\\ (x_2,st_2) \xleftarrow{\$} \mathcal{A}(st_1,\pi_{x_1});\\ \pi_{x_2} \leftarrow \mathsf{P}_2(CRS,ZSK,x_2);\\ \vdots\\ : CRS,tk,ZPK,x_1,\pi_{x_1},x_2,\pi_{x_2}\ldots\end{array}\right\}$$

- **Game Ideal.** In this game we consider the case of an adversary $\mathcal{A}$ interacting with an ideal prover SIM. SimSetup outputs a common reference string CRS' together with a trapdoor information tk. As in the real game, the adversary, after receiving (CRS',tk), chooses a database D, but now the simulator algorithm SimCom generates a "fake" public key ZPK' without the knowledge of D. Then the adversary adaptively chooses a sequence of elements $x_1,\ldots,x_n$ for which it wants to see the correct proofs. The queries are answered by SimProve that is assumed to have access to an oracle that, queried on a key x, returns the correct value associated to x in the database D if $x \in [D]$, otherwise it returns $\perp$. We stress about the fact that the simulator is allowed to query the database oracle only on the same keys queried by the adversary. The view of $\mathcal{A}$ in this game is:

$$View'(k) = \left\{\begin{array}{c}(CRS',tk) \xleftarrow{\$} \mathsf{SimSetup}(1^k);\\ (D,st) \xleftarrow{\$} \mathcal{A}(CRS',tk);\\ (ZPK',ZSK') \xleftarrow{\$} \mathsf{SimCom}(CRS',tk);\\ (x_1,st_1) \xleftarrow{\$} \mathcal{A}(st,ZPK');\\ \pi'_{x_1} \leftarrow \mathsf{SimProve}^{D(x_1)}(tk,ZSK',x_1):\\ (x_2,st_2) \xleftarrow{\$} \mathcal{A}(st_1,\pi'_{x_1});\\ \pi'_{x_2} \leftarrow \mathsf{SimProve}^{D(x_2)}(tk,ZSK',x_2);\\ \vdots\\ : CRS',tk,ZPK',x_1,\pi'_{x_1},x_2,\pi'_{x_2},\ldots\end{array}\right\}$$

Notice that we consider a notion of completeness that is less restrictive than the one suggested in [20]. Indeed, the definition given in [20] requires *perfect completeness*. Informally, such a requirement, prescribes that completeness is satisfied with probability 1 (rather than $1-\epsilon$, as in our case). We prefer to consider the less restrictive notion as it allows for more efficient solutions in practice.

III. Zero Knowledge EDB from trapdoor q-Mercurial Commitments

In this section we describe a construction of zero-knowledge EDB, from trapdoor q-mercurial commitments (defined in Section II-A), trapdoor mercurial commitments (see Appendix A) [4], [6] and collision resistant hash functions. The construction is very simple, as it generalizes easily from the original [6], [20] constructions. Still, it plays an important role in our quest for efficient zero knowledge sets, as it allows us to concentrate solely on the problem of realizing efficient qTMCs.

A. Intuitive construction

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

[6]Chase et al. [6] pointed out the necessity of quantifying only on efficiently computable databases in order to achieve computational ZK. For example a database may contain information needed to break the computational assumption used or to distinguish the generated transcript. In this case an adversary will be able to distinguish the real prover from the simulator.

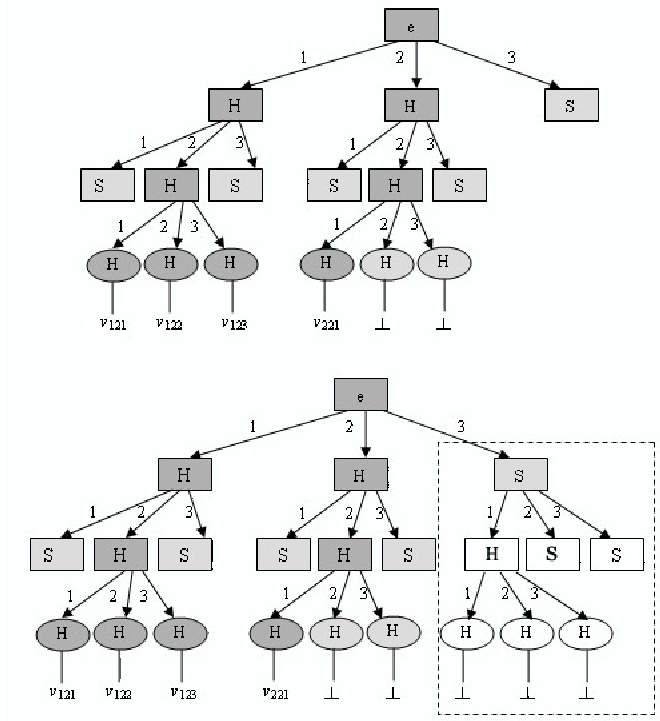

Fig. 1. **A 3-ary tree of height 3 before and after a query to the database key 311.** Each node of the tree contains a mercurial commitment: the label H is for hard commitments, S for the soft ones. Moreover the squares represent q-commitments, while the circles represent standard commitments. If the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

$D(x)$[7]. Each internal node contains the hard q-commitment to the hashes of the values contained in its q sons. The q-commitment contained in the root of the tree is the public key of the zero-knowledge EDB.

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

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

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

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

A detailed description of the construction sketched above, together with a complete security proof, is given in the following section.

B. Detailed construction

Here we give the details of our construction of zero-knowledge EDB from trapdoor q-mercurial commitments (see Section II-A), trapdoor mercurial commitments (Appendix A) [4], [6] and collision resistant hash functions.

Let T_k be the complete q-ary tree of height k, with q^k leaves. Let $\mathcal{U}_k$ be a universe of size q^k (i.e. $\{1, \cdots, q\}^k$). We build its associated tree T_k by labelling its nodes with the q-ary encoding of the elements $x \in \mathcal{U}_k$, so that all elements of $\mathcal{U}_k$ are leaves in T_k. Thus ϵ is the label for the root. If v is a non-leaf node, then $v1, \cdots, vq$ are its sons. If $S \subseteq \mathcal{U}_k$ we consider two subsets of T_k: $TREE(S)$ and $FRONTIER(S)$. $TREE(S)$ is the subtree of T_k containing all the nodes in the paths from the leaves in S to the root (the darker nodes in Figure 1). $FRONTIER(S) = \{v : v \notin TREE(S) \wedge parent(v) \in TREE(S)\}$ (the light shaded nodes in Figure 1).

We show our construction by describing its algorithms $(\mathsf{Setup}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$.

$\mathsf{Setup}(1^k)$. Given a security parameter k, it constructs the common reference string CRS as follows:

- it generates a pair of (matching) keys $(PK, TK) = \mathsf{qKeyGen}(1^k)$ for a trapdoor q-mercurial commitment scheme $\mathcal{QC}$;
- it generates a pair of (matching) keys $(PKM, TKM) = \mathsf{KeyGen}(1^k)$ for a trapdoor mercurial commitment scheme $\mathcal{C}$;
- it chooses a collision-resistant hash function H;
- finally it sets $CRS = (PK, PKM, H)$ and $tk = (TK, TKM)$.

THE DATABASE COMMITTER $\mathsf{P}_1(CRS, D)$.

1) Let S be the output space of H, when computed over the support $[D]$ of D, (i.e. $S = H([D])$). First, the committer constructs the tree $T = TREE(S) \bigcup FRONTIER(S)$.
2) $\forall$ leaf-nodes $H(x)$ it sets:
$$n_{H(x)} = \begin{cases} H(y) & \text{if } D(x) = y, \\ 0 & \text{if } D(x) = \emptyset. \end{cases}$$
It computes a hard commitment $(C_{H(x)}, \mathsf{aux}_{H(x)}) = \mathsf{HCom}_{PKM}(n_{H(x)})$ and sets $m_{H(x)} = H(C_{H(x)})$.
3) $\forall$ internal nodes u such that $u \in FRONTIER(S)$: $(C_u, \mathsf{aux}_u) = \mathsf{qSCom}_{PK}()$.
4) $\forall$ internal nodes $u \in TREE(S)$: $(C_u, \mathsf{aux}_u) = \mathsf{qHCom}_{PK}(m_{u1}, \cdots, m_{uq})$.
5) It stores in each node u the aux_u elements produced in the steps before.

6) $\forall$ internal nodes $u \in T \backslash \{\epsilon\} : m_u = H(C_u)$.
7) The committer outputs the commitment of the root as public key: $ZPK = (C_\epsilon)$. The secret key ZSK contains all the stored elements.

THE DATABASE PROVER $\mathsf{P}_2(CRS, ZSK, x)$. The prover algorithm produces a proof π_x of the database value of x. If u, v are two nodes of the tree such that v is a son of u then we write $i = index_u(v)$ to refer the position $i \in \{1, \cdots, q\}$ of v among the sons of u. 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:

$$\Big\{y, C_{H(x)}, HO_{H(x)} = \mathsf{HOpen}_{PKM}(H(y), \mathtt{aux}_{H(x)}), \{C_u, qHO_u = \mathsf{qHOpen}_{PK}(m_v, i, \mathtt{aux}_u)\}_{v=H(x), parent(x), \cdots, \epsilon-1}\Big\}$$

with $u = parent(v), i = index_u(v)$.
2) $\mathbf{D}(\mathbf{x}) = \emptyset$. The prover first checks if the leaf node $H(x)$ is in the tree T constructed by P_1. 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 P_1. The proof π_x contains the commitments and the soft openings in the path from $H(x)$ toward the root ϵ:

$$\Big\{C_{H(x)}, SO_{H(x)} = \mathsf{SOpen}_{PKM}(0, \mathbb{S}, \mathtt{aux}_{H(x)}), \{C_u, qSO_u = \mathsf{qSOpen}_{PK}(m_v, i, \mathbb{H}/\mathbb{S}, \mathtt{aux}_u)\}_{v=H(x), parent(x), \cdots, \epsilon-1}\Big\}$$

with $u = parent(v), i = index_u(v)$.

THE DATABASE VERIFIER $\mathsf{V}(CRS, ZPK, x, \pi_x)$. We consider two cases (depending on the type of proof received):

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) $D(x) = \emptyset$.
   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 $\emptyset$, else output $\perp$.

1) Proof of security: The following theorem proves that the scheme proposed above is a zero-knowledge EDB. The proof may be seen as an extension of the one given in [6] (that supports binary trees and mercurial commitments) to the case of q-ary trees and qTMCs.

Theorem 1: Assuming that $\mathcal{QC}$ is a trapdoor q-mercurial commitment scheme, $\mathcal{C}$ is a trapdoor mercurial commitment scheme and $\mathcal{H}$ is a family of collision resistant hash functions, the scheme $(\mathsf{Setup}, \mathsf{P}_1, \mathsf{P}_2, \mathsf{V})$ presented above is a zero-knowledge EDB.

Proof: To prove the theorem we prove separately that the scheme satisfies completeness, soundness and the zero-knowledge requirement.

Completeness. We show that completeness is satisfied with overwhelming probability $1 - \epsilon(k)$. Such $\epsilon(k)$ comes from the probability of finding a collision in the hash function H. It is easy to check that all tests made by the verifier algorithm are valid for the construction of P_1 and P_2 and the completeness of both the commitment schemes $\mathcal{C}$ and $\mathcal{QC}$.

Soundness. We assume there exists an adversary $\mathcal{A}$ that breaks the soundness of our scheme with non-negligible advantage ϵ. Then we can build a simulator $\mathcal{B}$ that, with non-negligible advantage, uses $\mathcal{A}$ to break either the mercurial binding of the mercurial commitment scheme or the q-mercurial binding of the q-mercurial commitment scheme or the collision resistance of the hash function.

$\mathcal{B}$ receives in input:

- the public key PKM for a mercurial commitment scheme;
- the public key PK for a q-mercurial commitment scheme;
- the description of a hash function H.

$\mathcal{B}$ constructs the common reference string $CRS = (PK, PKM, H)$ from the received values and gives it to $\mathcal{A}$. With probability ϵ the adversary outputs a tuple (ZPK', x, π_x, π'_x) such that

$$\mathsf{V}(CRS, ZPK', x, \pi_x), \mathsf{V}(CRS, ZPK', x, \pi'_x) \neq \perp$$

and $\mathsf{V}(CRS, ZPK', x, \pi_x) \neq \mathsf{V}(CRS, ZPK', x, \pi'_x)$.

Notice that the proofs π_x and π'_x have to be different (as they prove different statements). Still they must be both valid. Thus the two proofs have to be the same close to the root, but they must "fork" at some point. This leads to the following, mutually exclusive cases:

1) the commitments are the same but where the forking occurs, they are opened to different values. More specifically we distinguish between two subcases, one for the leaves and another for internal nodes:
   a) in the leaf-nodes: $C_{H(x)} = C'_{H(x)} \wedge (y \neq y' \vee \pi_x$ is of type $D(x) = y$ and π'_x of type $D(x) = \emptyset)$. In such case the simulator can break either the collision resistance of the hash function (if $H(y) = H(y')$) or the mercurial binding of the mercurial commitment scheme $\mathcal{C}$.
   b) for some internal node u: $m_{ui} \neq m'_{ui} \wedge C_u = C'_u$ and π_u is a hard opening for C_u to (m_{ui}, i) and π'_u

is a hard or soft opening for C_u to (m'_{ui}, i). In this case we have a hard or soft collision that breaks the q-mercurial binding property of the q-mercurial commitment scheme $\mathcal{QC}$.

2) Where the forking occurs, the commitments are the same, they open to identical values, but the messages were originally different. This means that we found a collision in the hash function:
   a) in the leaves: $y \neq y' \wedge H(y) = H(y')$,
   b) for some internal node u: $C_u \neq C'_u \wedge m_u = H(C_u) = H(C'_u) = m'_u$;

Zero-Knowledge. In order to show that our generic construction satisfies the zero-knowledge property we first describe a simulator $\mathsf{SIM} = (\mathsf{SimSetup}, \mathsf{SimCom}, \mathsf{SimProve})$ for the ideal game.

SimSetup is exactly the same as the real Setup algorithm. SimCom runs $(C_\epsilon, \mathtt{aux}_\epsilon) = \mathsf{qFake}_{PK,TK}()$ and sets $ZPK = C_\epsilon, ZSK = \mathtt{aux}_\epsilon$.

In order to simulate the first proof for a queried element x, the SimProve algorithm proceeds as follows:

1) it queries the database oracle on x to obtain the correct value $D(x)$;
2) it sets
$$n_{H(x)} = \begin{cases} H(y) & \text{if } D(x) = y, \\ 0 & \text{if } D(x) = \emptyset; \end{cases}$$
3) it makes a fake commitment $C_{H(x)}$ and sets $m_{H(x)} = H(C_{H(x)})$;
4) it creates a fake-commitment for each of the $q-1$ siblings of $H(x)$ and then computes the corresponding hashes;
5) it creates a fake q-commitment for the parent node of the q values generated in the two steps before;
6) it creates $q-1$ q-fake-commitments, one for each sibling of the node created in the step before;
7) it repeats step 5,6 until the root.

The proof π_x contains the hard (respectively soft) equivocations of the commitments in the path from $H(x)$ to the root node. Moreover SimCom updates the secret key with the values of the newly generated nodes.

When SimProve receives subsequent queries, it searches for the last node w in the path from the root to $H(x)$ present in the currently build tree. Then it works as above to build the nonexistent subtree rooted in w. As in the previous case, the proof π_x will contain hard (or soft) equivocations of the commitments in the path from $H(x)$ to the root.

Once we have defined the simulator for the ideal game we will prove that this game is indistinguishable from the real one as follows. Let us call GR the Game Real and GI the Game Ideal played by our simulator. We recall that to formally prove the zero-knowledge property we have to show that for all adversaries $\mathcal{A}$ the view produced by $\mathcal{A}$ in Game Real is indistinguishable from the one generated by $\mathcal{A}$ in Game Ideal (for brevity we write it as $GR \approx GI$). We prove this with a fairly standard hybrid argument where we define intermediate games in which we change step-by-step our simulator in the way it produces commitments and openings until it will behave like a real prover. Next we show that the difference between two such subsequent games can be reduced to the equivocation properties of the commitment scheme. Finally, since the number of games is at most polynomial, we will obtain the indistinguishability of the two main games GR and GI.

Given a database D, let Q be the number of nodes in the tree $T = TREE(S) \bigcup FRONTIER(S)$ where S is defined as before (i.e. $S = H([D])$). Assuming a level-ordering of T, for $i = 1, \ldots, Q$ we define T_i to be the tree containing the first i nodes of T (i.e. $T_0 = \emptyset$ and $T_Q = T$). Now consider the ideal game GI and define Game G_i to be GI modified such that SimCom receives in input T_i and both SimCom and SimProve proceed as follows. All the commitments (and openings) of all the tree nodes revealed by T_i are created as in the case of a real prover (i.e. by using the real algorithms). Instead the remaining ones contain commitments created using the fake algorithms. Then, starting from G_Q we can define games $G_{Q,j}$ for $j = 1, \ldots, d$ where d is the number of keys in D. The game $G_{Q,j}$ is the same as G_Q except that the simulator is also given the database values of the first j keys in D. Clearly we have that $GI = G_0$ and $GR = G_{Q,d}$.

Now we prove the following lemma to show that the views of two adjacent games are indistinguishable, namely $G_i \approx G_{i+1}\ \forall i = 0, \ldots, Q-1$ [8]. More formally, let $\mathcal{D}$ be an algorithm that takes in input a view generated by a run of an adversary $\mathcal{A}$ in our games and outputs 0 or 1 (we write it as $\mathcal{D}(G_i^{\mathcal{A}})$).

Lemma 1: If $\mathcal{QC}$ is a trapdoor q-mercurial commitment scheme, then for all adversaries $\mathcal{A}$ it holds:

$$|Pr[\mathcal{D}(G_i^{\mathcal{A}}) = 1] - Pr[\mathcal{D}(G_{i+1}^{\mathcal{A}}) = 1]| \leq \epsilon(k)$$

where ϵ is a negligible function.

Proof:

Notice that for a fixed database D and a set of adversary's queries $x_1, \ldots, x_n$ the two games G_i and G_{i+1} differ only in a specific node u: for the algorithms (the real ones or simulator's) used to create the commitment C_u and to produce the related openings.

We show how to convert an efficient distinguisher $\mathcal{D}$ into an algorithm $\mathcal{B}$ that breaks one of the equivocation properties of the commitment scheme.

Given a database D we distinguish for the node u between two possible cases such that at least one of them occurs with probability at least 1/2:

1) in the real game u would contain a soft commitment,
2) in the real game u would contain an hard commitment.

In the first case a distinguisher $\mathcal{D}$ for the two games G_i and G_{i+1} can be reduced to an adversary for the q-SEquivocation while in the second case we can make a reduction to q-HEquivocation.

[8] The lemma takes into account only the case of q-mercurial commitments. However it is easy to see that the same argument can be extended to standard mercurial commitments for the remaining d games concerning the last level of the tree.

Before starting the simulation $\mathcal{B}$ makes a random guess about one of the two cases and then runs the appropriate simulation as follows.

CASE 1. In this case $\mathcal{B}$ has guessed that u will contain a soft commitment and it plays the q-SEquivocation game. Thus $\mathcal{B}$ receives a pair of keys (PK, TK) for the q-mercurial commitment scheme and a commitment C. It generates a pair of keys (PKM, TKM) for the standard mercurial commitment and proceeds as follows. Operations for all nodes $v \neq u$ are simulated using (PK, TK) and the appropriate algorithms according to their position in the tree. When $\mathcal{B}$ gets to create C_u, if (according to D) it has to be an hard commitment, $\mathcal{B}$ aborts and outputs a random bit. Otherwise it sets $C_u = C$. Then $\mathcal{B}$ generates $C_{u1}, \ldots, C_{uq}$ and gives them to its challenger to get back a soft opening of C_u for each of these values. Such openings will be later used by $\mathcal{B}$ to produce those proofs that "pass through" u. It is easy to see that if the q-SEquivocation challenger has chosen $b = 1$ (i.e. C is fake), then $\mathcal{B}$ is simulating game G_i, otherwise it simulates G_{i+1}.

Let V be the view produced in this simulation. At the end $\mathcal{B}$ runs $b' \leftarrow \mathcal{D}(V)$ and outputs the same b'.

CASE 2. In this game $\mathcal{B}$ has guessed that u will contain an hard commitment and it plays the q-HEquivocation game. So $\mathcal{B}$ receives a pair of keys (PK, TK) for the q-mercurial commitment scheme. It also generates a pair of keys (PKM, TKM) for the standard mercurial commitment and proceeds as follows. Operations for all nodes $v \neq u$ are simulated using PK, TK and the appropriate algorithms according to their position in the tree. When $\mathcal{B}$ gets to create C_u, it first generates $C_{u1}, \ldots, C_{uq}$ using the fake algorithm $\mathsf{qFake}_{PK,TK}()$ and gives these values to its challenger. $\mathcal{B}$ gets back a commitment C and the hard and soft openings $\pi_{u1}, \tau_{u1}, \ldots, \pi_{uq}, \tau_{uq}$ of C to $C_{u1}, \ldots, C_{uq}$ respectively. It sets $C_u = C$ and uses the received openings to produce those proofs that "pass through" u. It is easy to see that if the q-HEquivocation challenger has chosen $b = 1$, then $\mathcal{B}$ is simulating game G_i, otherwise it simulates G_{i+1}.

Let V be the view produced in this simulation. When the game is over $\mathcal{B}$ runs $b' \leftarrow \mathcal{D}(V)$ and outputs the same bit b'.

In conclusion we have that

$$\left| Pr[\mathcal{D}(G_i^{\mathcal{A}}) = 1] - Pr[\mathcal{D}(G_{i+1}^{\mathcal{A}}) = 1] \right| \leq \frac{1}{2} Adv^{S-Equiv}(\mathcal{B}) + \frac{1}{2} Adv^{H-Equiv}(\mathcal{B})$$

■

Applying the result of Lemma 1 to all the games defined before we finally have that $\left| Pr[\mathcal{D}(GR^{\mathcal{A}})] - Pr[\mathcal{D}(GI^{\mathcal{A}})] \right|$ is negligible.

■

IV. TRAPDOOR q-MERCURIAL COMMITMENT BASED ON SDH

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

Our construction relies on the Strong Diffie-Hellman assumption (SDH for short), introduced by Boneh and Boyen in [3]. Informally, the SDH assumption in bilinear groups G_1, G_2 of prime order p states that, for every PPT algorithm $\mathcal{A}$ and for a parameter q, the following probability is negligible:

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

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

The SDH assumption obviously implies the discrete logarithm assumption (i.e. if the former holds, so has to do the latter). A reduction in the other direction, however, is not known. Recently, however, Cheon [7] proved that, for many primes p, the q-Strong Diffie Hellman problem has computational complexity reduced by $O(\sqrt{q})$ with respect to that of the discrete logarithm problem (in the same group).

THE NEW SCHEME. Now we describe our proposed trapdoor q-Mercurial Commitment scheme, in terms of its component algorithms $(\mathsf{qKeyGen}, \mathsf{qHCom}, \mathsf{qHOpen}, \mathsf{qHVer}, \mathsf{qSCom}, \mathsf{qSOpen}, \mathsf{qSVer}, \mathsf{qFake}, \mathsf{qHEquiv}, \mathsf{qSEquiv})$, as defined in Section II-A.

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

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

- $\mathsf{qKeyGen}(1^k, q)$: the key generation algorithm runs a bilinear group generator $\mathcal{G}(1^k)$ for which the SDH assumption holds [3] to get back the description of groups G_1, G_2, G_T and a bilinear map $e : G_1 \times G_2 \rightarrow G_T$. Such groups share the same prime order p.
  The description of the groups contains the group generators: $g_1 \in G_1$, $g_2 \in G_2$. The algorithm proceeds by picking a random integer $x \leftarrow \mathbb{Z}_p^*$ and it sets $A_1 = g_1^x, \cdots, A_q = g_1^{x^q}, h = g_2^x$. Next, it chooses a collision resistant hash function H from $\mathcal{H}$.
  The public key is set as $PK = (g_1, A_1, \cdots, A_q, g_2, h, H)$, while the trapdoor is $TK = x$:
- $\mathsf{qHCom}_{PK}(m_1, \cdots, m_q)$: the hard commitment algorithm randomly selects $\alpha, w \leftarrow \mathbb{Z}_p^*$ and computes $C_i = H(i||m_i), \forall i = 1, \cdots, q$ (the symbol $||$ denotes concatenation). Next, it defines the polynomial

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

  and sets $g_1' = (\prod_{i=0}^{q} A_i^{\alpha^i \beta_i})^w = g_1^{f(\alpha x)w}$ and $g_2' = h^\alpha$. In the unlucky case that either $g_1' = 1$ or $g_2' = 1$, then one simply retries with another random α.
  Thus, letting $\gamma = \alpha x$, we have $g_1' = g_1^{f(\gamma)w}$ and $g_2' = h^\alpha = g_2^\gamma$.
  The commitment is (g_1', g_2'). The auxiliary information is $\mathsf{aux} = (\alpha, w, m_1, \cdots, m_q)$;
- $\mathsf{qHOpen}_{PK}(m, j, \mathsf{aux})$ outputs $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$;

- $\mathsf{qHVer}_{PK}(m, j, C, \pi)$ computes the $q-1$ terms $C_i = H(i||m_i)\ \forall m_i \in \pi$ and $C_j = H(j||m)$. Next, it defines the polynomial $f(z) = \prod_{i=1}^{q}(z + C_i)$ and computes the β_i coefficients as above.
  Checks if $g'_1 = (\prod_{i=0}^{q} A_i^{\alpha^i \beta_i})^w$ and $g'_2 = h^\alpha$. If both tests succeed, it outputs 1;
- $\mathsf{qSCom}_{PK}()$: picks $\alpha', y \leftarrow \mathbb{Z}_p^*$ at random, sets
$$g'_1 = g_1^{\alpha'},\ \ g'_2 = g_2^y$$
and outputs (g'_1, g'_2) and $\mathsf{aux} = (\alpha', y)$;
- $\mathsf{qSOpen}_{PK}(m, j, \mathsf{flag}, \mathsf{aux})$: if $\mathsf{flag} = \mathbb{H}$ the algorithm computes $C_i = H(i||m_i), \forall i = 1, \cdots, j-1, j+1, \cdots, q$, $C_j = H(j||m)$, and sets
$$f_j(z) = \frac{f(z)}{(z + C_j)} = \prod_{i=1 \wedge i \neq j}^{q} (z + C_i) = \sum_{i=0}^{q-1} \delta_i z^i$$
Next, it computes $\sigma_j = (\prod_{i=0}^{q-1} A_i^{\delta_i \alpha^i})^w = g_1^{\frac{f(\gamma)w}{\gamma + C_j}} = (g'_1)^{\frac{1}{\gamma + C_j}}$. The output is σ_j.
  If $\mathsf{flag} = \mathbb{S}$ the algorithm computes $C_j = H(j||m)$ and outputs $\sigma_j = (g'_1)^{\frac{1}{y + C_j}}$;
- $\mathsf{qSVer}_{PK}(m, j, C, \tau)$: the soft verification algorithm takes in input a message m and an index $j \in \{1, \cdots, q\}$. It computes $C_j = H(j||m_j)$, and checks if
$$e(\sigma_j, g'_2 g_2^{C_j}) = e(g'_1, g_2).$$
  If this is the case, it outputs 1;
- $\mathsf{qFake}_{PK,TK}()$: the fake commitment algorithm is the same as qSCom;
- $\mathsf{qHEquiv}_{PK,TK}(m_1, \cdots, m_q, j, \mathsf{aux})$: the non-adaptive hard equivocation algorithm uses the trapdoor key TK to hard open a fake commitment (which is originally a commitment to nothing). It computes $C_i = H(i||m_i), \forall i = 1, \cdots, q$ and constructs the polynomial
$$f(z) = \prod_{i=1}^{q}(z + C_i) = \sum_{i=0}^{q} \beta_i z^i.$$
  It sets $\alpha = \frac{y}{x}, w = \frac{\alpha'}{f(y)}$ and outputs $\pi = \{\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q\}$;
- $\mathsf{qSEquiv}_{PK,TK}(m, j, \mathsf{aux})$: the soft equivocation algorithm is the same as qSOpen.

A. Properties of the scheme

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

Theorem 2 (Security of qTMC*):* Assuming that the Strong Diffie-Hellmann holds for $\mathcal{G}$ and $\mathcal{H}$ is a family of collision resistant hash functions, then $\mathcal{QC}$ is a trapdoor q-mercurial commitment scheme.

Proof: To prove the theorem we need to make sure that the proposed scheme is binding and hiding, in the sense discussed in Section II-A. We prove each property separately.

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

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

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

$\mathcal{B}^{\mathbb{S}}$ receives in input from its challenger a $(q+3)$-tuple $(g_1, g_1^x, \cdots, g_1^{x^q}, g_2, g_2^x)$ and the description of a hash function H. The simulator runs $\mathcal{A}^{\mathbb{S}}$ on input such values as the public key of the q-mercurial commitment scheme. Then with probability ϵ the adversary outputs (C, m, j, π, m', τ) such that: $C = (g'_1, g'_2)$ is a commitment, $m \neq m'$, $\pi = (\alpha, w, m_1, \cdots, m_{j-1}, m_{j+1}, \cdots, m_q)$ is a valid hard opening for C to the message m at position j and $\tau = (\sigma_j)$ is a valid soft opening for C to m' of index j. We distinguish two cases:

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

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

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

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

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

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

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

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

2) $m \neq m'$ and $C_j \neq C'_j$.

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

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

As observed in section II-A, if a scheme is proper, q-HEquivocation can be simplified by giving to the adversary only the hard openings for the messages it has chosen. Therefore, since our scheme has this property, we can apply such simplification to our proof. In both cases we will show that it is infeasible for an adversary to distinguish between a real commitment/decommitment tuple from a fake/equivocation one.

In the q-HEquivocation game the adversary is asked to tell apart

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

from

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

It is easy to see that in both cases α, w are uniformly random in $\mathbb{Z}_p^*$ and follow the same distribution. In the first tuple, they are chosen uniformly and at random, while in the second tuple they are distributed, respectively, as y and α', which were chosen uniformly and at random in $\mathbb{Z}_p^*$. Thus the two distributions are indistinguishable.

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

$$\{(g_1^{\alpha'}, g_2^{y}), \{\sigma_i = (g'_1)^{\frac{1}{\gamma + C_i}}\}_{i=1,\ldots,q}\}$$

$$\{(g_1^{\alpha'}, g_2^{y}), \{\sigma_i = (g'_1)^{\frac{1}{\gamma + C_i}}\}_{i=1,\ldots,q}\}$$

are distributed in exactly the same manner.

■

V. Efficiency

In the previous section we proposed a trapdoor q-mercurial commitment scheme $\mathcal{QC}$ based on the Strong Diffie-Hellmann assumption. In order to build efficient zero knowledge EDB, we also use a trapdoor mercurial commitment scheme $\mathcal{C}$ based on the Discrete Logarithm constructions given in [6], [20] (the interested reader is deferred to Appendix B for a quick description of such a scheme). For our convenience we consider an implementation of the scheme that allows us to use some of the parameter already in use for the qTMC scheme. In particular, we use $g_1, A_1 \in G_1$ from the public key of $\mathcal{QC}$ as the public key for $\mathcal{C}$.

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

In the following we propose a formal analysis of the space required by the proofs in our ZK-EDB construction and we compare it with the best known solution. To obtain indications about the time efficiency of the available schemes, a series of experimental tests on real implementations have been carried on. The resulting measures are reported later.

A. Space analysis

Below we compare our proposal with the most efficient (in terms of space) implementation known so far, namely the one by Micali *et al.* [20] (MRK from now on, for short), when implemented over elliptic curves with short representation (the best already known solution in terms of space).

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

1) Groups used in the comparisons: Following [11] we fix a security parameter $\ell = 256$ to achieve $k = 128$ bits of security. Specifically 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*, G_2 is a subgroup of $E(\mathbb{F}_{p^e})$ and $G_T \subset E(\mathbb{F}^*_{p^e})$. In particular we consider elliptic curves with embedding degree $e = 12$ and CM discriminant $D = -3$. As suggested in [11], for the case of *Type 3 groups* (see [11] for details), such parameters enable to obtain elements of G_2 that have size twice the size of elements of G_1.

2) Bandwidth analysis: A proof of membership in our scheme contains: 1 mercurial hard-commitment and the related hard-opening for the leaf (we do not count the eventual associated value) together with $(h-1)$ q-mercurial hard-commitments and h q-mercurial hard-openings (one for each level above the leaves, but the commitment is not necessary for the root). In this analysis we count the space in term of elements[9] of G_1: a mercurial hard-commitment, as well as the related hard-opening, requires 2 elements, a q-mercurial hard-commitment counts as 3 elements and the related hard-opening requires 2 elements with additional $(q-1)$ elements for the hash values of the sibling commitments. In this way a proof of membership in our scheme requires $h(q+4)+1$ elements. The proof of non-membership has the same kind of components but in the "soft" version: 1 mercurial soft-commitment (2 elements), 1 mercurial soft-opening (1 element), $(h-1)$ q-mercurial soft-commitment (3 elements for each) and h q-mercurial soft-opening (1 element for each). The total is $4h$ elements.

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

q	Membership	Non-membership
2	769	512
4	**513**	256
8	**513**	170.7
16	641	128
32	922.6	102.4
64	1451.7	85.3
128	2414.7	73.1
256	4161	**64**

TABLE I
SPACE REQUIRED BY PROOFS, IN OUR SCHEME (BEST VALUES IN BOLD)

In MRK's scheme a proof of membership includes 2 mercurial hard-commitments (one for the node on the path and one for the brother) for each level different than the root (so k levels) and 1 mercurial hard-opening for each possible level (so $k+1$). The total is $6k+2$ elements. The "soft" versions of such components in a proof of non-membership require a total of $5k+1$ elements.

In both the schemes each element has size ℓ but, for our construction, we let q vary. For such a choice of parameters we obtain the following results.

The scheme of Micali *et al.* requires 770 elements for proofs of membership and 641 for proofs of non-membership. Results for our scheme are summarized in Table I.

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

Notice that such a choice of q (i.e. $q = 8$) keeps the scheme practical also in terms of length of the common reference string. Notice also that, according to our present knowledge of the SDH problem, it seems reasonable to consider the same security parameter for our scheme and for the MRK implementation. This is because Cheon [7] attack requires q to be an upper-bound to a factor of either $p-1$ or $p+1$ in order to be effective. If one sets $q = 8$, as suggested in the table above, this would imply that one should increase the key size of at most 2 bits in the worst case. Thus using the same security parameter for both ours and MRK seems to be reasonable for all practical purposes.

B. Computational experiments

For the sake of completeness and to get indications about the time efficiency, a series of computational experiments have been carried out. We consider three possible EDB constructions: the original MRK construction on a subgroup of residue classes of integers modulo a safe prime (MRK-MOD, in the follow), the same construction over elliptic curves (MRK-ECC, for short) and our construction using qTMC over the same elliptic curves.

1) Implementation details: The schemes have been implemented in language C using the GNU-MP v.4.2.4 and PBC v.0.4.18 libraries [13], [25]. All the codes share the same level of optimization, using pre-computation to speed-up the operations where it is possible[10]. As in the previous analysis, the chosen security parameter is $\ell = 256$ to achieve $k = 128$ bits of security. As recommended in [23], the MRK-MOD scheme uses a subgroup of order 256 bits with element representations of 3072 bits and the other two schemes use a subgroup of points on an elliptic curve of order 256 bits[11]. SHA-256 is used as collision resistant hash function. The machine used for tests has an Intel Core 2 Duo CPU at 2.4GHz.

2) The test results: The goal of a first experiment is to study the time efficiency of the construction based on qTMC using different values for q. During the tests an EDB with 200 elements with keys of 120 bits is committed and a series of queries (both for belonging and for non-belonging elements) and verifications of the resulting proofs are executed. The measures (in time and space) are reported in Table II. For the space occupation of the proofs, the optimal value $q = 8$ determined through the previous formal analysis is clearly confirmed as a good choice. Such value looks a good compromise also for the time efficiency: it is optimal in several measures and quite good in the rest.

The goal of a second experiment is to compare (in time and space) the three considered schemes: the value $q = 8$ is adopted in the construction based on qTMC. The results are summarized in Table III. The MRK-MOD is confirmed as the worst choice from almost all the points of view: its time efficiency is probably compromised by the large modulus required to satisfy the security level. Our proposal, in spite of the more complex underling structure, is computationally more efficient than the original MRK-MOD (but not than MRK-ECC). As stated before, it beats the competitors in the space efficiency. On the other hand, we have to note that the verification of non-membership proofs is quite slow in our construction: this is due to the pairing applications involved in that step[12].

VI. CONCLUSIONS

In this paper we introduced and implemented the notion of trapdoor q mercurial commitments. Our construction can be used to construct Zero-Knowledge Sets that allow for proofs that are much shorter than those obtained by previous work. It would be interesting to investigate if it is possible to come up with an even more efficient implementation of the new primitive. In particular, it would be very interesting to construct qTMCs that allow for openings whose length is independent of q.

[10]The sources of the tests are available upon request to the authors.

[11]In the PBC library, the required elliptic curves is the "type F" that uses groups of order 256 bits, with elements representations of 256 bits for G_1 and of 512 bits for G_2. To be more precise, the representation requires a further single bit to store the sign of the y component of the point.

[12]We notice that this might be improved with a better support by the PBC library for the pre-computation in the pairing function applications.

measure	$q = 2$	$q = 4$	$q = 8$	$q = 16$	$q = 32$	$q = 64$
commit EDB (ms)	140196.762	**104850.553**	116795.299	157833.864	238242.889	387052.189
query on $\in$ elem. (ms)	0.940	0.500	0.280	0.280	**0.220**	**0.220**
verify of proofs $\in$ (ms)	657.401	390.584	**343.521**	381.224	503.951	755.447
query on $\notin$ elem. (ms)	1261.899	**870.074**	905.977	1160.653	1699.826	2720.550
verify of proofs $\notin$ (ms)	14708.679	7364.700	4897.366	3678.130	2996.627	**2634.925**
space $\in$ proofs (bits)	185056	123376	**123296**	153976	221536	348496
space $\notin$ proofs (bits)	123240	61620	41080	30810	24648	**20540**

TABLE II
MEASURES ON OUR EDB CONSTRUCTION WITH DIFFERENT VALUES FOR q (BEST VALUES IN BOLD)

measure	MRK-MOD	MRK-ECC	our scheme ($q = 8$)
commit EDB (ms)	545470.090	**144977.060**	326308.393
query on $\in$ elem. (ms)	**0.168**	0.200	0.360
verify of proofs $\in$ (ms)	895.456	539.122	**359.350**
query on $\notin$ elem. (ms)	1066.827	**285.154**	934.858
verify of proofs $\notin$ (ms)	661.001	**478.174**	5172.419
space $\in$ proofs (bits)	1613312	194552	**129448**
space $\notin$ proofs (bits)	1580800	162040	**43134**

TABLE III
COMPARISON OF THE EFFICIENCY OF THE THREE SCHEMES (BEST VALUES IN BOLD)

REFERENCES

[1] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *ACM Conference on Computer and Communications Security*, 1993.

[2] M. Blum, A. De Santis, S. Micali and P. Persiano. Non Interactive Zero Knowledge. *SIAM Journal on Computing*, 20(6), 1991.

[3] D. Boneh and X. Boyen. Short Signatures Without Random Oracles. *Advances in Cryptology – proceedings of EUROCRYPT 2004*, pages 56–73, LNCS 3027, 2004.

[4] D. Catalano, Y. Dodis and I. Visconti. Mercurial Commitments: Minimal Assumptions and Efficient Constructions. *Theory of Cryptography Conference – TCC 2006*, pages 120–144, 2006.

[5] D. Catalano, D. Fiore, and M. Messina. Zero-knowledge sets with short proofs. *Advances in Cryptology – proceedings of EUROCRYPT 2008*, LNCS 4965, pages 433–450, 2008.

[6] M. Chase, A. Healy, A. Lysyanskaya, T. Malkin and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. *Advances in Cryptology – proceedings of EUROCRYPT 2005*, LNCS 3494, pages 422–439, 2005.

[7] J.H. Cheon. Security Analysis of the Strong Diffie-Hellman Problem. *Advances in Cryptology – proceedings of EUROCRYPT 2006*, LNCS 4004, pages 1–11, 2006.

[8] R. Cramer and I. Damgård. New Generation of Secure and Practical RSA-based signatures. *Advances in Cryptology – proceedings of CRYPTO '96*, LNCS 1109, pages 173–185, 1996.

[9] I. Damgård. Collision free hash functions and public key signature schemes. *Advances in Cryptology – proceedings of EUROCRYPT '87*, LNCS 304, pages 203–216, 1987.

[10] C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. *Journal of Cryptology*, 11(3) 1998, pages 187–208.

[11] S.D. Galbraith, K.G. Paterson and N.P. Smart. Pairings for Cryptographers. *Cryptology ePrint Archive, Report 2006/165*, 2006. `http://eprint.iacr.org`.

[12] R. Gennaro and S. Micali. Independent Zero-Knowledge Sets. *Proceedings of ICALP 2006*, pages 34–45, 2006.

[13] GNU Multiple Precision Arithmetic Library. `http://www.gmplib.org`.

[14] S. Goldwasser and R. Ostrovsky. Invariant Signatures and Non Interactive Zero Knowledge proofs are equivalent. *Advances in Cryptology – proceedings of CRYPTO '92*, pages 228–245, 1993.

[15] B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. *Proceedings of TCC 2010*, pages 499–517, 2010.

[16] C.H. Lim. Efficient Multi-Exponentiation and Application to Batch Verification of Digital Signatures. *Unpublished manuscript*, August 2000.

[17] C.H. Lim. and P.J. Lee. More Flexible Exponentiation with Precomputation. *Advances in Cryptology – proceedings of CRYPTO '94*, LNCS 839, pages 95–107, 1994.

[18] M. Liskov. Updatable zero-knowledge databases. *Advances in Cryptology – proceedings of ASIACRYPT 2005*, LNCS 3788, pages 174-198, 2005.

[19] R. Merkle. A Digital Signature based on a Conventional Encryption Function. *Advances in Cryptology – proceedings of CRYPTO '87*, LNCS 293, pages 369–378, 1988.

[20] S. Micali, M. Rabin and J.K. Kilian. Zero-Knowledge Sets. *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science – FOCS '03*, page 80, 2003.

[21] S. Micali, M. Rabin and S. Vadhan. Verifiable Random Functions. *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science – FOCS '99*, page 120, 1999.

[22] L. Nguyen. Accumulators from Bilinear Pairings and Applications. *Topics in Cryptology – CT-RSA 2005*, LNCS 3376, pages 275–292, 2005.

[23] NIST Recommendation for Key Management, 2005.

[24] R. Ostrovsky, C. Rackoff and A. Smith. Efficient Consistency Proofs for Generalized Queries on a Committed Database. *Proceedings of ICALP 2004*, LNCS 3142, pages 3–26, 2004.

[25] PBC (Pairing-Based Cryptography) Library. `http://crypto.stanford.edu/pbc/`.

[26] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Advances in Cryptology – proceedings of CRYPTO '91*, LNCS 567, pages 129–140, 1992.

[27] M. Prabhakaran and R. Xue. Statistically Hiding Sets. *Cryptology ePrint Archive, Report 2007/349*, 2007. `http://eprint.iacr.org`.

APPENDIX A
TRAPDOOR MERCURIAL COMMITMENTS

A *trapdoor mercurial commitment scheme* is defined by the following set of algorithms:
$(\mathsf{KeyGen}, \mathsf{HCom}, \mathsf{HOpen}, \mathsf{HVer}, \mathsf{SCom}, \mathsf{SOpen}, \mathsf{SVer}, \mathsf{Fake}, \mathsf{HEquiv}, \mathsf{SEquiv})$.

- $\mathsf{KeyGen}(1^k)$: 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;

- $\mathsf{HOpen}_{pk}(m, \mathtt{aux})$: the hard opening algorithm produces a hard decommitment π to the message m correlated to $(C, \mathtt{aux}) = \mathsf{HCom}_{pk}(m)$;
- $\mathsf{HVer}_{pk}(m, C, \pi)$: 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;
- $\mathsf{SOpen}_{pk}(m, \mathsf{flag}, \mathtt{aux})$: 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 $(C, \mathtt{aux}) = \mathsf{HCom}_{pk}(m)$ or to a soft commitment $(C, \mathtt{aux}) = \mathsf{SCom}_{pk}()$. 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 $(C, \mathtt{aux}) = \mathsf{Fake}_{pk,tk}()$:

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

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

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.
  - **HEquivocation.** $\mathcal{A}$ chooses $m \in \mathcal{M}$ and gives it to the challenger. If $b = 0$ the challenger gives to $\mathcal{A}$ a tuple (C, π, τ) such that: $(C, \mathtt{aux}) = \mathsf{HCom}_{pk}(m)$, $\pi = \mathsf{HOpen}_{pk}(m, \mathtt{aux})$ and $\tau = \mathsf{SOpen}_{pk}(m, \mathbb{H}, \mathtt{aux})$. Otherwise it returns (C, π, τ) such that: $(C, \mathtt{aux}) = \mathsf{Fake}_{pk,tk}()$, $\pi = \mathsf{HEquiv}_{pk,tk}(m, \mathtt{aux})$ and $\tau = \mathsf{SEquiv}_{pk,tk}(m, \mathtt{aux})$.
  - **SEquivocation.** If $b = 0$ the challenger generates $(C, \mathtt{aux}) = \mathsf{SCom}_{pk}()$ and gives C to $\mathcal{A}$. Then $\mathcal{A}$ chooses $m \in \mathcal{M}$, gives such m to the challenger and receives $\mathsf{SOpen}_{pk}(m, \mathbb{S}, \mathtt{aux})$. 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 $\mathsf{SEquiv}_{pk,tk}(m, \mathtt{aux})$.

  At the end of the game $\mathcal{A}$ outputs b' as its guess for b and wins if $b' = b$.

  It is easy to see that the mercurial hiding is implied by the HEquivocation. Moreover if the scheme is proper (i.e. the soft opening is a proper subset of the hard one), then the HEquivocation can be simplified by giving to the adversary only the hard opening π.

RELATION TO PREVIOUS DEFINITIONS. We observe that the definition of equivocations stated above is slightly different from the one in [4] where the authors define three equivocation properties: HHEquivocation, HSEquivocation and SSEquivocation. They claim that a mercurial commitment scheme that satisfies their definition satisfies also the one of Chase *et al.* [6]. The statement is true but what is not made clear in their paper is that, in order to make the proof valid, the scheme has to be proper. From a theoretical point of view the claim still holds for all schemes, as a non-proper mercurial commitment can be alway converted into a proper one: simply by defining the hard opening as the hard opening plus the soft opening. However this may be tricky in practice. First of all because if this is not said clearly one may use a non-proper scheme not being aware that it does not enjoy itself (i.e. without applying the conversion to proper) all the properties guaranteed by the Chase *et al.*'s definition (e.g. building ZKS). Second, if one has an efficient non-proper scheme, the conversion to proper may leak something in efficiency.

Therefore in our work we propose a slightly different definition that is more general than the one of Catalano *et al.* [4]. Our SEquivocation is the same as the SSEquivocation

in [4] while our HEquivocation is like the HHEquivocation with the addition that we allow the adversary to receive also the soft opening of the challenge commitment. It is easy to see that our new definition implies the definition of Chase *et al.* [6] without any requirement on the properness of the scheme and thus it also allows to build ZKS directly (i.e. without any conversion) from *any* mercurial commitment scheme (proper and non-proper). Finally it still has the same nice property of the definition in [4] of being easy to prove, compared with the one in [6].

APPENDIX B
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 [20], and later formalized by Chase *et al.* in [6].

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;
- $\mathsf{HOpen}_{pk}(m, \mathtt{aux})$: produces a hard decommitment π which contains the auxiliary information $\mathtt{aux} = r, s$;
- $\mathsf{HVer}_{pk}((C, h_r), m, \pi)$: 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 $\mathbb{Z}_p^*$;
- $\mathsf{SOpen}_{pk}(m, \mathtt{aux})$: produces a soft decommitment $\tau = s'$ to the message m for a commitment produces using $\mathtt{aux}$. 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;
- $\mathsf{HEquiv}_{pk,tk}(m, \mathtt{aux})$: uses the trapdoor key to create a valid hard decommitment $\pi = r, s'$ to m where $s' = s - m/rx$;
- $\mathsf{SEquiv}_{pk,tk}(m, \mathtt{aux})$: returns a soft decommitment $\tau = s'$, with s' computed as in the hard equivocation.