Zero-Knowledge Elementary Databases with More Expressive Queries

Benoît Libert[1,2], Khoa Nguyen[3], Benjamin Hong Meng Tan[3,4], and Huaxiong Wang[3]

[1] CNRS, Laboratoire LIP, France
[2] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France
[3] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
[4] Institute for Infocomm Research, A*STAR, Singapore

Abstract. Zero-knowledge elementary databases (ZK-EDBs) are cryptographic schemes that allow a prover to commit to a set D of key-value pairs so as to be able to prove statements such as "x belongs to the support of D and $\mathsf{D}(x) = y$" or "x is not in the support of D". Importantly, proofs should leak no information beyond the proven statement and even the size of D should remain private. Chase *et al.* (Eurocrypt'05) showed that ZK-EDBs are implied by a special flavor of non-interactive commitment, called *mercurial commitment*, which enables efficient instantiations based on standard number theoretic assumptions. On the other hand, the resulting ZK-EDBs are only known to support proofs for simple statements like (non-)membership and value assignments. In this paper, we show that mercurial commitments actually enable significantly richer queries. We show that, modulo an additional security property met by all known efficient constructions, they actually enable range queries over keys and values – even for ranges of super-polynomial size – as well as membership/non-membership queries over the space of values. Beyond that, we exploit the range queries to realize richer queries such as k-nearest neighbors and revealing the k smallest or largest records within a given range. In addition, we provide a new realization of trapdoor mercurial commitment from standard lattice asssumptions, thus obtaining the most expressive quantum-safe ZK-EDB construction so far.

Keywords. Zero-knowledge databases, expressive queries, lattice-based commitments.

1 Introduction

Zero-knowledge sets (ZKS), as introduced by Micali, Rabin and Kilian [21], allow a prover P to commit to a finite set S without revealing its size. The commitment is generated such that the prover can efficiently and non-interactively prove the membership or non-membership of certain elements x in the committed set S. The zero-knowledge property mandates that proofs reveal no information beyond the truth of the statement: even its cardinality should remain hidden. The soundness property captures the prover's inability to prove contradictory statements "$x \in S$" and "$x \notin S$" about the same S.

Zero-knowledge elementary databases (ZK-EDBs) generalize the notion of zero-knowledge sets to elementary databases (EDBs). An EDB D is a partial function: a set of key-value pairs (x, y) where each key x of the universe occurs at most once and thus takes at most one value $y = \mathsf{D}(x)$. For syntactic reasons, keys x not in D are assigned $\mathsf{D}(x) = \perp$. Each query x obtains a response $\mathsf{D}(x)$ and a proof of its correctness. Again, proofs should reveal no information beyond the value $\mathsf{D}(x)$: particularly the number of records in D. Here, soundness requires the infeasibility of proving two distinct values y, y' for any given x. Micali *et al.* [21] described an elegant construction of ZK-EDB based on the discrete logarithm assumption, which was generalized by Chase *et al.* [5,6] to a general design of ZK-EDBs from a lower-level primitive called *mercurial commitment*.

In short, mercurial commitments are commitment schemes which generate commitments in either a hard or soft mode. The former satisfies the usual binding property while the latter allows the sender to create dummy commitments that do not commit the sender to any message. The ZK-EDB constructions of [21,5,6] combine mercurial commitments with a Merkle tree [20], where each internal node contains a mercurial commitment to its two children. The existence of dummy commitments is exactly what allows the sender to commit to the database in polynomial time without revealing its size. The latter is hidden by having a super-polynomial upper bound on the number of leaves in the Merkle tree. Each leaf is assigned to a key x and contains a real commitment to the value $y = \mathsf{D}(x)$ and every internal node contains a commitment to its two children. By storing a dummy commitment at the root of each empty subtree, the sender is able to commit to the entire $\mathsf{D} = \{(x, y)\}$ in polynomial time.

While efficient and based on standard assumptions, the ZK-EDB realizations of [21,5,6] have relatively limited expressivity; only simple statements like "x does not belong in D" or "x is in D with value $y = \mathsf{D}(x)$" can be proved. In this paper, we show that mercurial commitments actually enable proofs of more involved statements like range queries over keys and values as well as k-nearest neighbour and k-minimum/maximum queries. As special cases, our techniques make it possible to prove membership or non-membership over values, which was not known to be possible without revealing the database size.

Our contribution. In this paper, we investigate the extent to which expressive queries can be proven with efficient ZK-EDB protocols from mercurial commitments. We extend the constructions of [21,5,6] to allow the prover to convincingly answer queries of the form "Give me all database records $(x, y) \in \mathsf{D}$ whose keys x lie within the range $[a_x, b_x]$". For any $[a_x, b_x]$ of super-polynomial length, we show that a simple tweak in mercurial commitments allows efficient, polynomial-sized proofs of correctness of the response without leaking the database size.

In a second step, we extend this technique so as to handle range queries over values. Namely, for a super-polynomially large interval $[a_y, b_y]$, we allow the prover to answer queries "Send me all records $(x, y) \in \mathsf{D}$ with values y in the interval $[a_y, b_y]$". Again, we can prove correctness of the response in zero-knowledge with a polynomial-size proof. As a special case of range queries over values, we can prove statements like "No key x of the database is assigned the value y" or "y occurs in D and the corresponding set of keys is $\mathsf{D}^{-1}(y)$". We note that previous ZK-EDB protocols [21,5,6] were unable to handle such statements while hiding the database size: the only way to prove that no record of the form $(*, y)$ exists was to prove inequalities $y_i \neq y$ for all records (x_i, y_i).

In a third step, we also handle range queries over records. Namely, each query consists of a "narrow" rectangle $[a_x, b_x] \times [a_y, b_y]$ and the response consists of all records (x, y) such that $x \in [a_x, b_x]$ and $y \in [a_y, b_y]$. Here, we can handle rectangles of polynomial width $[a_y, b_y]$ and super-polynomial height $[a_x, b_x]$ with a proof size which is linear in the size of $[a_y, b_y]$ and the number of records in $[a_x, b_x] \times [a_y, b_y]$. However, the proof length does not depend on $(b_x - a_x)$, allowing it to be very large. As a special case $[x, x] \times [y, y]$ of range query over records, we can efficiently prove that specific records (x, y) do not belong to D, which amounts to saying "if x is in D at all, the corresponding value is not y". In the full version of this paper, we apply range queries to enable more interesting queries such as k-nearest neighbour and k-minimim/maximum. In the following, we refer to ZK-EDB protocols supporting such richer queries as "Zero-knowledge expressive elementary database" (ZK-EEDB).

We insist on building ZK-EEDBs without interaction or random oracles: as in [21,5,6], only a common reference string is assumed, which is necessary for NIZK proofs in the standard model anyway [1]. Our constructions are instantiable with existing mercurial commitments based on standard number theoretic assumptions. We identify a new equivocation property of mercurial commitments which is actually present in a generic construction of trapdoor mercurial commitment from Σ-protocols due to Catalano *et al.* [2]. Since the number theoretic constructions of [21,5,6] can be seen as instantiations of the general construction of [2], this immediately provides us with ZK-EEDBs based on the discrete logarithm and factoring/RSA assumptions. In addition, we provide a new construction of trapdoor mercurial commitment (TMC) based on a well-studied assumption in standard (i.e., non-ideal) lattices. Our new lattice-based TMC is a direct construction, which is not implied by the generic construction of [2]; rather, it draws inspiration from [21]. In non-ideal lattices, it performs better than TMC schemes implied by [2] under the same assumptions.

Our Techniques. Our setting involves a database owner who publishes a short string com_{D} that commits him to a particular database D consisting of records, which are key-value pairs $(x, \mathsf{D}(x))$, where $x, \mathsf{D}(x) \in [0, 2^\ell)$. The prover is required to answer queries and prove that the response is consistent with the committed database D in zero-knowledge, including not revealing how many keys x are in the support $[\mathsf{D}]$ of D. For this purpose, we follow the approach of using mercurial commitments [5,6].

In mercurial commitments, the binding property is relaxed by allowing the committer to softly open a commitment and say "The commitment opens to this message if it can be opened at all". During the commitment phase, the sender can either create a hard commitment, which can be hard/soft-opened to a unique message, or a soft commitment, which it can soft-open to any message. Unlike soft commitments, hard commitments can be opened both in the soft and the hard way, but soft openings can never contradict hard ones. Besides, hard and soft commitments should be computationally indistinguishable.

When a Merkle tree has a super-polynomial number of leaves, the prover has to store a soft commitment at the root of each empty sub-tree in order to commit to an EDB in polynomial time. In order to prove that some key is not in the database, the prover can soft-open all soft commitments on the path that connects the corresponding leaf to the root while generating the missing soft commitments at the time of proving non-membership.

When it comes to generating a proof for a range query $[a_x, b_x]$ over keys, the difficulty is to find a way to convince a verifier that no key of $[a_x, b_x]$ was omitted in the response. If $[a_x, b_x]$ is super-polynomially large, we cannot generate proofs of non-membership for all elements of $[a_x, b_x]$ that are not in the support $[\mathsf{D}]$ of D. Our solution to this problem is to rely on the Subset Cover framework of Naor, Naor and Lotspiech [25] and find the smallest set of nodes $\mathcal{P}$ that contains an ancestor of all leaves $[a_x, b_x] \setminus [\mathsf{D}]$ and no ancestor of those in $[a_x, b_x] \cap [\mathsf{D}]$. For each node $x \in \mathcal{P}$, we can have the prover convince the verifier that the soft commitment associated to x (which is created if it did not exist yet and authenticated via a path from x to the root) is really a soft commitment, by revealing the soft-commitment coins. For the sake of proving the zero-knowledge property, we need that the simulator be able to create fake commitments which can be subsequently equivocated by revealing fake hard/soft openings or pretending that they were soft commitments. For this purpose, we thus define a new equivocation property of mercurial commitments by requiring that fake commitments be not only equivocable as defined by prior works [2], but also "explainable" as soft commitments by using a trapdoor to compute plausible soft commitment coins. Fortunately, all known trapdoor mercurial commitments based on standard assumptions [2,5] satisfy this additional equivocation property. By using the Complete Subtree technique of Naor *et al.* [25], we are able to prove range queries $[a_x, b_x]$ in zero-knowledge with proofs of size $O(\ell \cdot |\mathfrak{R}| \cdot \log(b_x - a_x))$, where $\mathfrak{R} = [a_x, b_x] \cap [\mathsf{D}]$ and ℓ is the height of the Merkle tree.

In order to handle range queries over values, our idea is to have the prover commit to D by generating two Merkle trees. While the first one is computed in the same way as in ordinary ZK-EDBs, the second tree is used as a "reversed database" D^{-1}: namely, the keys of D^{-1} are the values y of D and their values are ZKS commitments to all the keys $x \in \mathsf{D}^{-1}(y)$ such that $(x, y) \in \mathsf{D}$. The reversed database D^{-1} thus uses nested Merkle trees in that each leaf y of D^{-1} may be assigned a value $com_{\mathsf{D}_y^{-1}}$, which is itself a size-hiding Merkle tree commitment whose non-empty leaves contain the keys x of D that map to y. Of course, we need to prevent the prover from cheating by using inconsistent Merkle trees in the two commitments com_{D} and $com_{\mathsf{D}^{-1}}$. To this end, we thus have proofs of membership consist of authentication paths in the two Merkle trees. By doing so, we can show that no dishonest prover can prove contradictory statements without breaking the binding property of the mercurial commitment scheme.

Our NIZK proofs for range queries readily carry over to prove the correctness of responses to range queries over values $[a_y, b_y]$. In particular, it yields a simple method of proving that a given value is not reached by the partial function D.

Our lattice-based trapdoor mercurial commitment is statistically hiding and computationally binding under the Short-Integer-Solution (SIS) assumption [24]. It builds on the lattice-based trapdoor commitment (KTX) of Kawachi *et al.* [16] and Micciancio-Peikert trapdoors [22]. While partially inspired by the discrete-log-based construction of [5], it is a direct construction with large message space which is *not* implied by the generic constructions of [5,6,2].

Intuitively, we generate two public matrices $\mathbf{A}_0, \mathbf{A}_1$, the former to be applied to messages and the latter to determine the mode of the commitment. When producing a commitment to some message, using a random matrix $\mathbf{R}$, we first compute a matrix $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1]$, where $\mathbf{B}_1 = \mathbf{A}_1\mathbf{R}$ (resp. $\mathbf{B}_1 = \mathbf{G} - \mathbf{A}_1\mathbf{R}$) if the commitment is a hard (resp. soft) one. The pair $\mathbf{A}_0, \mathbf{B}$ can be considered the public key of an instance of the KTX commitment scheme, with an associated trapdoor for $\mathbf{B}$ if the mercurial commitment is a soft one.

A mercurial commitment to a message, $\boldsymbol{\mu}$, is a commitment, "public key" pair, $\mathbf{C} = (\mathbf{c} = \mathbf{A}\boldsymbol{\mu} + \mathbf{B}\mathbf{r}, \mathbf{B}_1)$ for some commitment randomness $\mathbf{r}$. The two flavors of openings are straightforward: Soft openings to $\boldsymbol{\mu}$ are simply openings of $\mathbf{c}$ to $\boldsymbol{\mu}$ with the associated "public key" $\mathbf{A}_0, \mathbf{B} = [\mathbf{A} \mid \mathbf{B}_1]$. Hard openings, on the other hand, have an additional step of showing that $\mathbf{B}_1 = \mathbf{A}_1\mathbf{R}$ for some $\mathbf{R}$, essentially demonstrating that the "public key" does not have an embedded trapdoor.

Catalano *et al.* [2, Section 5] built a TMC scheme with large message space from any trapdoor commitment where a Σ-protocol allows proving knowledge of an opening to 0. For this purpose, the Σ-protocol is required to have a large challenge space, which becomes the message space of the TMC scheme. In the lattice setting, the only known Σ-protocols [19] with large challenge space operate over ideal lattices and thus require less standard assumptions than non-ideal lattices. Moreover, their honest-verifier zero-knowledge property relies on the prover performing rejection sampling and outputting a simulated transcript only with some probability, say $1/c$, for some constant c. Since the TMC scheme of [2, Section 5] generates hard commitments by running the HVZK simulator of the underlying Σ-protocol, the hard-committer can only produce a properly distributed hard commitment after c attempts on average. Our TMC scheme eliminates the need for several attempts and only requires one attempt to generate a hard commitment.

Related Work. Ostrovsky, Rackoff and Smith [28] described protocols handling orthogonal multi-dimensional range queries for committed databases allowing for d-dimensional key spaces. While their protocols extend to provide privacy by means of zero-knowledge proofs, they do not hide the database size. Chase *et al.* [5,6] and Catalano *et al.* [2] described size-hiding constructions of ZK-EDBs under general assumptions. In particular, Catalano, Dodis and Visconti [2] gave simplified security definitions for (trapdoor) mercurial commitments and showed how to obtain them from one-way functions in the shared random string model.

An EDB D is a partial function: a set of key-value pairs (x, y) where each key x of the universe occurs at most once and thus takes at most one value $y = \mathsf{D}(x)$.

Liskov [18] considered the notion of updatable zero-knowledge databases in the random oracle model. Prabhakaran and Xue [31] put forth the similar notion of statistically hiding sets, which allows for more efficient constructions. For the sake of efficiency, Kate *et al.* [15] considered quasi-database commitments which do not aim at hiding the database size. Catalano, Fiore and Messina [4] suggested a technique for compressing proofs of non-membership in ZK-EDB protocols. Libert and Yung [17] extended their idea to compress both proofs of membership and non-membership, while Catalano and Fiore [3] achieved similar proof compressions under more standard number theoretic assumptions.

An orthogonal line of work investigated the feasibility of stronger definitions in size-hiding database commitments. Gennaro and Micali [9] formalized the notion of independent ZK-EDBs, which prevents adversaries from correlating their committed databases to those of honest committers. In the plain model, Chase and Visconti [7] considered zero-knowledge protocols providing stronger simulation-based security at the expense of an interactive commitment phase.

The aforementioned constructions all relate to elementary databases. Ghosh *et al.* [11] formalized the notion of zero-knowledge lists. In the random oracle model, they gave size-hiding protocols where the prover can demonstrate the order in which elements appear in a committed list. Goyal *et al.* [13] gave black-box constructions of size-hiding database commitments supporting more general queries. Their goal is orthogonal to ours as they rely on the “MPC-in-the-head” technique [14] to obtain black-box constructions using interaction. Here, we aim at non-interactive constructions in the standard model from standard assumptions, although we restrict ourselves to range queries.

We also mention a large body of work devoted to authenticated data structures [26,32,30,29,27,12]. We insist that these result address a different problem than ours as they stand in the three party setting. Namely, in order to achieve a better efficiency, they assume that the committer is a honest database owner that always faithfully computes commitments whereas proofs are generated by an untrusted server. While reasonable in some applications (e.g., certificate revocation with a trusted certification authority [26]), the assumption of a honest committer is too much to ask for in other settings. With a pricing database, for example, it is desirable to have guarantees against price discrimination by the database owner. For this reason, we focus on the two-party setting which is usually more challenging and results in less efficient schemes. Our protocols are indeed less efficient than the range queries of Ghosh *et al.* [12] – which, to our knowledge, is the best size-hiding construction handling range queries in the three-party setting – but they do not assume a trusted committer.

2 Preliminaries

Notations. In our notations, λ always stands for the security parameter. Let ϵ denote the empty string. For $x \in \{0,1\}^{\ell}$, let x' be the binary string that is equal to x except with the final bit flipped and $x0$ be ($x1$ respectively) the string of length $\ell + 1$ with 0 (1 respectively) appended to x. Besides that, we denote the string consisting of the first i bits of x with $x|_i$. For a string of length ℓ, $x|_0 = \epsilon$ and $x|_\ell = x$. For a set $\mathcal{S}$, $U(\mathcal{S})$ denotes the uniform distribution over $\mathcal{S}$ and $x \leftarrow U(\mathcal{S})$ means that element x is sampled from the distribution $U(\mathcal{S})$.

For another elementary database $\mathsf{D} = \{(x, \mathsf{D}(x))\} \subset [0, 2^\ell) \times [0, 2^\ell)$, a set of key-value pairs, let $[\mathsf{D}]$ denote the set of keys $x \in [0, 2^\ell)$ such that there exists a $y \in [0, 2^\ell)$ with $(x, y) \in \mathsf{D}$. We write $\mathsf{D}(x) = \perp$ to indicate that there exists no $y \in [0, 2^\ell)$ such that $(x, y) \in \mathsf{D}$. We write $x \in \mathsf{D}$ to say that $(x, \mathsf{D}(x)) \in \mathsf{D}$ for some $\mathsf{D}(x) \in [0, 2^\ell)$, if there is no ambiguity. For a range $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$, we use $[\mathfrak{R}]$ to denote $[a_x, b_x]$.

2.1 Trapdoor Mercurial Commitments

Informally, trapdoor mercurial commitments (TMC) are commitment schemes with two flavors of commitments and openings: hard and soft. Hard commitments are like regular commitments to a message M and can only be hard- and soft-opened to M. Hard openings are like regular openings for hard commitments. Soft commitments commit to no particular message and cannot be hard-opened at all but can be soft-opened to any message. Soft openings tease that a commitment potentially opens to some message M, and corresponds to the statement "if this commitment can be hard-opened at all, it can only be to M". Following the definitions proposed by Catalano, Dodis and Visconti [2], TMC consists of ten PPT algorithms, (Setup, $\mathbb{H}\mathsf{Commit}$, $\mathbb{H}\mathsf{Open}$, $\mathbb{H}\mathsf{Verify}$, $\mathbb{S}\mathsf{Commit}$, $\mathbb{S}\mathsf{Open}$, $\mathbb{S}\mathsf{Verify}$, MFake, $\mathbb{H}\mathsf{Equivocate}, \mathbb{S}\mathsf{Equivocate}$).

- $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$: Taking security parameter λ as input, outputs a public mercurial commitment key mpk and secret mercurial trapdoor msk.
- $C \leftarrow \mathbb{H}\mathsf{Commit}(mpk, M; R)$: Taking public key mpk, message M and random coins R as inputs, outputs a hard commitment C for M.
- $\pi \leftarrow \mathbb{H}\mathsf{Open}(mpk, M; R)$: Taking public key mpk, message M and random coins R as inputs, outputs a hard opening π for C of M.
- $\mathbb{H}\mathsf{Verify}(mpk, M, C, \pi)$: Taking public key mpk, message M, commitment C and hard opening π as inputs, accepts if π proves that C is a valid hard commitment to M and rejects otherwise.
- $C \leftarrow \mathbb{S}\mathsf{Commit}(mpk; R)$: Taking public key mpk and random coins R as inputs, output a soft commitment C to no message in particular.
- $\tau \leftarrow \mathbb{S}\mathsf{Open}(mpk, M, \mathsf{flag}; R)$: Given mpk, M, a flag flag and random coins R, if $\mathsf{flag} = \mathbb{H}$, output soft opening τ "associated" to hard commitment $C = \mathbb{H}\mathsf{Commit}(mpk, M; R)$. Otherwise, $\mathsf{flag} = \mathbb{S}$ and τ is a soft opening "associated" to the soft commitment $C = \mathbb{S}\mathsf{Commit}(mpk; R)$ for message M.
- $\mathbb{S}\mathsf{Verify}(mpk, M, C, \tau)$: Taking public key mpk, message M, commitment C and soft opening τ, accepts if C can be potentially hard opened to M in the future and rejects otherwise.
- $C \leftarrow \mathsf{MFake}(msk; R)$: Taking secret key msk and random coins R as inputs, outputs a "fake" commitment C that are initially not tied to any message.

- $\pi \leftarrow \mathbb{H}\mathsf{Equivocate}(msk, M; R)$: Taking secret key msk, message M and random coins R, outputs a supposedly valid hard opening π (*hard-fake*) of the fake commitment $C = \mathsf{MFake}(msk; R)$ to M.
- $\tau \leftarrow \mathbb{S}\mathsf{Equivocate}(msk, M; R)$: Taking secret key msk, message M and random coins R, outputs a supposedly valid soft opening τ (*soft-fake*) of the fake commitment $C = \mathsf{MFake}(msk; R)$.

Remark 1. In many cases, including all currently known constructions, the soft opening of a hard commitment is a proper part of the hard opening to the same message. Therefore, $\mathbb{S}\mathsf{Verify}$ performs a proper subset of the tests done by $\mathbb{H}\mathsf{Verify}$. Such trapdoor mercurial commitment schemes are called *proper*.

Correctness. Trapdoor mercurial commitments are *correct* if, with overwhelming probability, for all $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, and message space $\mathcal{M}$

- Hard commitments: For all messages $M \in \mathcal{M}$ and for all random coins R, if $C = \mathbb{H}\mathsf{Commit}(mpk, M; R)$, then
  1. for all $\tau \leftarrow \mathbb{S}\mathsf{Open}(mpk, M, \mathbb{H}; R)$, $\mathbb{S}\mathsf{Verify}(mpk, M, C, \tau)$ accepts.
  2. for all $\pi \leftarrow \mathbb{H}\mathsf{Open}(mpk, M; R)$, $\mathbb{H}\mathsf{Verify}(mpk, M, C, \pi)$ accepts.
- Soft commitments: For all coins R, if $C \leftarrow \mathbb{S}\mathsf{Commit}(mpk; R)$, then for all $M \in \mathcal{M}$ and $\tau \leftarrow \mathbb{S}\mathsf{Open}(mpk, M, \mathbb{S}; R)$, $\mathbb{S}\mathsf{Verify}(mpk, M, C, \tau)$ accepts.
- Equivocations: For all random coins R, if $C \leftarrow \mathsf{MFake}(msk; R)$, then for all $M \in \mathcal{M}$, the following conditions are satisfied w.h.p.
  1. If $\pi \leftarrow \mathbb{H}\mathsf{Equivocate}(msk, M; R)$, $\mathbb{H}\mathsf{Verify}(mpk, M, C, \pi)$ accepts.
  2. If $\tau \leftarrow \mathbb{S}\mathsf{Equivocate}(msk, M; R)$, $\mathbb{S}\mathsf{Verify}(mpk, M, C, \tau)$ accepts.
  3. If $R' \leftarrow \mathsf{FakeExplain}(msk, R)$, we have $C = \mathbb{S}\mathsf{Commit}(mpk; R')$.

Security. The security properties are similar to trapdoor commitments, *binding*, *hiding* and *equivocation*, except they are modified to accommodate the two different flavors of commitments and openings.

- *Mercurial-binding*: Given mpk, no PPT adversary $\mathcal{A}$ can find C, M, π, M', π' (respectively C, M, τ, M', π') such that π (respectively τ) is a valid hard (respectively soft) opening of C to M and π' is a valid hard opening of C to $M \neq M'$.
- *Mercurial-hiding*: No PPT adversary $\mathcal{A}$, given mpk, can find a message $M \in \mathcal{M}$ where it can distinguish a random hard commitment/soft opening tuple $(M, \mathbb{H}\mathsf{Commit}(mpk, M; R), \mathbb{S}\mathsf{Open}(mpk, M, \mathbb{H}; R))$ from a random soft commitment/soft opening tuple $(M, \mathbb{S}\mathsf{Commit}(mpk; R), \mathbb{S}\mathsf{Open}(mpk, M, \mathbb{S}; R))$.

In particular, the mercurial-binding property implies that $\mathcal{A}$ cannot find C which can be soft-opened or hard-opened to one message and then hard-opened to another: a soft opening can never disagree with a hard opening. This also implies the infeasibility of hard opening a commitment C to some message and simultaneously explain it as a soft commitment.

Catalano *et al.* [2] formalized the hiding properties of trapdoor mercurial commitments with several equivocation properties. They require the existence of an algorithm producing fake commitments which can be equivocated in a hard and soft way using a trapdoor. Even if the trapdoor is public, it should be infeasible to distinguish fake commitments and their equivocations into hard (resp. soft) commitments from hard (resp. soft) commitments and their hard (resp. soft) openings. On top of these three equivocation properties, we introduce a 4-th property called Soft-Explain equivocation (or SE equivocation for short). Namely, the trapdoor msk should make it possible to explain a fake commitment by outputting plausible random coins that explain it as a soft commitment.

- *Equivocation*: There are three related conditions for equivocation that have to be satisfied by mercurial commitments. Each is defined by a pair of games, one real and one ideal, and no PPT adversary A can distinguish between them, even if the trapdoor key msk is given at the beginning of each game, real or ideal. In all games R denotes a set of random coins sampled from the appropriate distribution.
    - *HH Equivocation*: The real game has $\mathcal{A}$ choose a message $M \in \mathcal{M}$ and receive $(M,\ \mathbb{H}\mathsf{Commit}(mpk, M; R), \mathbb{H}\mathsf{Open}(mpk, M; R))$ while the ideal game has $\mathcal{A}$ choose a message $M \in \mathcal{M}$ and obtain the tuple $(M, \mathsf{MFake}(msk; R), \mathbb{H}\mathsf{Equivocate}(msk, M; R))$.
    - *HS Equivocation*: The real game has $\mathcal{A}$ choose a message $M \in \mathcal{M}$ and receive $(M,\ \mathbb{H}\mathsf{Commit}(mpk, M; R), \mathbb{S}\mathsf{Open}(mpk, M; R))$ while the ideal game has $\mathcal{A}$ choose a message $M \in \mathcal{M}$ and obtain the tuple $(M, \mathsf{MFake}(msk; R), \mathbb{S}\mathsf{Equivocate}(msk, M; R))$.
    - *SS Equivocation*: The real game has $\mathcal{A}$ first get $C = \mathbb{S}\mathsf{Commit}(mpk; R)$, then choose $M \in \mathcal{M}$ and finally receive $\mathbb{S}\mathsf{Open}(mpk, M, \mathbb{S}; R)$ while the ideal game has $\mathcal{A}$ first get $C = \mathsf{MFake}(msk; R)$, then choose $M \in \mathcal{M}$ and receive $\mathbb{S}\mathsf{Equivocate}(msk, M; R)$.

Remark 2. As noted by Catalano *et al.* [2], HS and SS equivocation implies mercurial-hiding. In addition, for proper mercurial commitments, HH equivocation implies HS equivocation. So it suffices to verify HH and SS equivocations and mercurial-binding for the security of any proper mercurial commitment scheme.

2.2 Merkle Trees

Let $\mathcal{T}_\ell$ denote a full and complete binary tree of depth ℓ, with the depth of the root defined as 0 and leaves ℓ. Nodes at depth $i > 0$ are labeled with i-bit binary strings corresponding to the i-bit binary decomposition of 0 to $2^i - 1$. Let $[a, b], [a, b)$ denote the set $\{a, a + 1, \ldots, b - 1, b\}$ and $\{a, a + 1, \ldots, b - 1\}$ respectively. For any node x in the tree $\mathcal{T}_\ell$, we let x' mean its sibling in the tree. We call the canonical covering of $[a, b]$, $\mathcal{P}_{[a,b]}$, the unique minimal set of nodes of $\mathcal{T}_\ell$ such that each node in $[a, b]$ is the descendant of some node in $\mathcal{P}_{[a,b]}$ and for every node in $x \in \mathcal{P}_{[a,b]}$, the subtree rooted at x has leaves that are all within $[a, b]$.

Zero-Knowledge Elementary Databases and Sets. Proposed by Micali, Rabin and Kilian [21], zero-knowledge elementary databases (ZK-EDB) and sets (ZKS) enable efficient answers to membership queries in zero-knowldege.

ZK-EDB is a scheme that allows one to commit to a secret database D of records and non-interactively produce proofs of (non-)membership. Membership queries on D committed in com_{D} take a key x as input and expect an answer $\big(x, \mathsf{D}(x)\big)$ which is the record in D corresponding to the key x if $x \in \mathsf{D}$. In particular, zero-knowledge sets are ZK-EDBs where $\mathsf{D}(x) = 1$ if $x \in \mathsf{D}$.

Formally, a ZK-EDB has four algorithms (Init, ComDB, ProveQ, $\mathsf{VerifyQ}$),

- $(crs, tk) \leftarrow \mathsf{Init}(1^\lambda)$: Taking security parameter λ as input, generates and outputs common reference string (CRS) crs and trapdoor information tk.
- $(com, \Delta) \leftarrow \mathsf{ComDB}(crs, \mathsf{D})$: Taking the CRS crs and database D as inputs, outputs a commitment of D, com, and opening information Δ.
- $\Pi_x \leftarrow \mathsf{ProveQ}(crs, (com, \Delta), x)$: Taking the CRS crs, database commitment and opening information (com, Δ) and key x as inputs, outputs a proof Π_x of either $x \in \mathsf{D}$ or $x \notin \mathsf{D}$.
- $y \leftarrow \mathsf{VerifyQ}(crs, com, x, \Pi_x)$: Taking the CRS crs, database commitment com, key x and proof Π_x as inputs, outputs y where

$$y = \begin{cases} \mathsf{D}(x), & \text{if } x \in [\mathsf{D}]; \\ \perp, & \text{if } x \notin [\mathsf{D}]; \\ bad, & \text{if it otherwise believes that the prover is cheating.} \end{cases}$$

Security. The three security properties of ZK-EDB are *completeness*, *soundness* and *zero-knowledge*. The first one requires that honestly generated proofs always satisfy verification with $\mathsf{VerifyQ}$. Soundness mandates that provers be unable to produce a key x and successful proofs Π_x, Π'_x such that they do not verify to the same value y. Finally, zero-knowledge implies that each proof Π_x only reveals the value $\mathsf{D}(x)$ and nothing else about D.

Merkle Trees from Trapdoor Mercurial Commitments. Although Micali *et al.* constructed ZKS and ZK-EDB specifically from number-theoretic assumptions, Chase *et al.* [5,6] introduced the TMC primitive and showed that ZKS and ZK-EDB are simply Merkle trees built with TMC. The key to their size-hiding property is that TMC allows a committer compute portions of the Merkle tree that do not contain database elements only when required in proofs.

We detail four algorithms, $\mathsf{BuildTree}$, $\mathbb{H}\mathsf{OpenPath}$, $\mathbb{S}\mathsf{OpenPath}$ and $\mathsf{VerifyPath}$, which we will use in Sections 3, 4. These algorithms encapsulate the construction of a ZK-EDB scheme from TMC in [5,6]: ComDB corresponds to $\mathsf{BuildTree}$, ProveQ to $\mathbb{H}\mathsf{OpenPath}$ and $\mathbb{S}\mathsf{OpenPath}$ based on the value $\mathsf{D}(x)$ and $\mathsf{VerifyQ}$ to $\mathsf{VerifyPath}$. Let λ be a security parameter, $(crs, tk) \leftarrow \mathsf{Setup}(1^\lambda)$ and a database $\mathsf{D} = \{(x, \mathsf{D}(x)) \mid (x, \mathsf{D}(x)) \in [0, 2^\ell) \times [0, 2^\ell)\}$.

- $(com, \Delta) \leftarrow \mathsf{BuildTree}(crs, \mathsf{D})$: Taking as inputs CRS crs and database D, build a Merkle tree of depth ℓ, indexed by strings in $\bigcup_{i=0}^{\ell} \{0,1\}^i$, as follows:
  1. For each leaf $j \in \{0,1\}^\ell$ with $\mathsf{D}(j) \neq \perp$, $C_j = \mathbb{H}\mathsf{Commit}(crs, \mathsf{D}(j); R_j)$. For every leaf j with its sibling $j' \in \mathsf{D}$ but $j \notin \mathsf{D}$, set $C_j = \mathbb{S}\mathsf{Commit}(crs; R_j)$. For all other leaves j, set $C_j = nil$.

2. At depth i from $\ell-1$ to 0 and each $\rho \in \{0,1\}^i$, define C_ρ as follows. For all ρ such that $C_{\rho 0}, C_{\rho 1} \neq nil$, set $C_\rho = \mathbb{H}\mathsf{Commit}(crs, (C_{\rho 0}, C_{\rho 1}); R_\rho)$. For all ρ such that C_ρ was defined but not $C_{\rho'}$, $C_{\rho'} = \mathbb{S}\mathsf{Commit}(crs; R_{\rho'})$. For any other string $\rho \in \{0,1\}^i$, set $C_\rho = nil$.
3. After the end of Step 2, if the value at the root $C_\epsilon = nil$, meaning $\{C_j\} = \emptyset$, then set $C_\epsilon = \mathbb{S}\mathsf{Commit}(crs; R_\epsilon)$.

Output $com = C_\epsilon$ and $\varDelta = \{R_j\}$, the set of random coins for all commitments computed in the steps above.

- $\varPi_z \leftarrow \mathbb{H}\mathsf{OpenPath}(crs, (com, \varDelta), z)$: Given crs, a database commitment com and the opening information $\varDelta$ for a database D and a key $z \in \mathsf{D}$, define the hard authentication path for $z \in \mathsf{D}$ as the set of hard openings for nodes in indices $z = z|_\ell, z|_{\ell-1}, \ldots, z|_1$ which form a path from z to the root $\epsilon = z|_0$. Proceed to decommit all the nodes on the path as follows:
  1. Compute $\pi_z \leftarrow \mathbb{H}\mathsf{Open}(crs, (z, \mathsf{D}(z)); R_z)$.
  2. At each depth j from $\ell-1$ to 0, compute the hard opening for $C_{z|_j}$ to $(C_{z|_j 0}, C_{z|_j 1})$, $\pi_{z|_j} \leftarrow \mathbb{H}\mathsf{Open}(crs, (C_{z|_j 0}, C_{z|_j 1}); R_{z|_j})$.

  Output $\varPi_z = (\mathsf{D}(z), \{C_{z|_j}, C_{(z|_j)'}\}_{1 \leq j \leq \ell}, \{\pi_{z|_j}\}_{0 \leq j \leq \ell})$.
- $\varPi_z \leftarrow \mathbb{S}\mathsf{OpenPath}(crs, (com, \varDelta), z)$: Taking as inputs CRS crs, database commitment com and opening information $\varDelta$ for a database D and a key $z \in \mathsf{D}$, define the soft authentication path for $z \notin \mathsf{D}$ as the set of soft openings for nodes at indices $z = z|_\ell, z|_{\ell-1}, \ldots, z|_1$ which form a path from z to the root $\epsilon = z|_0$. Let h be the largest value such that $C_{z|_h} \neq nil$.
  1. If the complete path does not exist, i.e., $C_z = nil$, fill it out to leaf z:
     a. Compute $C_z = \mathbb{S}\mathsf{Commit}(crs; R_z)$, $C_{z'} = \mathbb{S}\mathsf{Commit}(crs; R_{z'})$.
     b. At depth j from $\ell-1$ to $h+1$, compute $C_{z|_j} = \mathbb{S}\mathsf{Commit}(crs; R_{z|_j})$ and $C_{(z|_j)'} = \mathbb{S}\mathsf{Commit}(crs; R_{(z|_j)'})$.
  2. Otherwise, $C_z = \mathbb{S}\mathsf{Commit}(crs; R_z)$ and we proceed to the next step.
  3. Produce soft openings to nodes along the path from leaf z to the root.
     a. Compute $\tau_z = \mathbb{S}\mathsf{Open}(crs, \perp, \mathbb{S}; R_z)$, soft opening of C_z to $\perp$.
     b. At depth j from $\ell-1$ to $h+1$, compute soft openings of $C_{z|_j}$ to their children, $\tau_{z|_j} = \mathbb{S}\mathsf{Open}(crs, (C_{z|_j 0}, C_{z|_j 1}), \mathbb{S}; R_{z|_j})$.
     c. For j from h to 1, compute $\tau_{z|_h} = \mathbb{S}\mathsf{Open}(crs, (C_{z|_h 0}, C_{z|_h 1}), \mathbb{H}; R_{z|_h})$.
     d. If $C_\epsilon = \mathbb{S}\mathsf{Commit}(crs; R_\epsilon)$, set $\tau_\epsilon = \mathbb{S}\mathsf{Open}(crs, (C_{z|_0}, C_{z|_1}), \mathbb{S}; R_\epsilon)$. Otherwise, $\tau_\epsilon = \mathbb{S}\mathsf{Open}(crs, (C_{z|_0}, C_{z|_1}), \mathbb{H}; R_\epsilon)$

  Output $\varPi_z = (\perp, \{C_{z|_j}, C_{(z|_j)'}\}_{1 \leq j \leq \ell}, \{\tau_{z|_j}\}_{0 \leq j \leq \ell})$. Also, add any random coins used when a path is filled out to $\varDelta$ for use with later proofs.
- $ans \leftarrow \mathsf{VerifyPath}(crs, com, z, \varPi_z)$: Taking as inputs CRS crs, database commitment com, key z and proof $\varPi_z$, check the proof which has two possible forms:
  - $\mathsf{D}(z) \neq \perp$: $\varPi_z = (\mathsf{D}(z), \{C_{z|_j}, C_{(z|_j)'}\}_{1 \leq j \leq \ell}, \{\pi_{z|_j}\}_{0 \leq j \leq \ell})$.
    1. Run $\mathbb{H}\mathsf{Verify}(crs, \mathsf{D}(z), C_z, \pi_z)$ and set $ans = bad$ if it rejects.
    2. Otherwise, for j from $\ell-1$ to 0, run $\mathbb{H}\mathsf{Verify}(crs, (C_{z|_j 0}, C_{z_j 1}, \pi_{z|_j})$ and set $ans = bad$ if any of them reject.

If $ans \neq bad$, then set and output $ans = \mathsf{D}(z)$.

- $\mathsf{D}(z) = \perp$: $\Pi_z = (\perp, \tau_z, \{C_{z|_j}, C_{(z|_j)'}\}_{1 \leq j \leq \ell}, \{\tau_{z|_j}\}_{0 \leq j \leq \ell})$.
  1. Run $\mathbb{S}\mathsf{Verify}(crs, \perp, C_z, \tau_z)$ and set $ans = bad$ if it does not accept.
  2. Otherwise, for j from $\ell - 1$ to 0, run $\mathbb{S}\mathsf{Verify}(crs, (C_{z|_j 0}, C_{z_j 1}, \tau_{z|_j})$ and set $ans = bad$ if any of them do not accept.

  If $ans \neq bad$, then set and output $ans = \perp$.

Complete Subtree Method. We recall the complete subtree method proposed by Naor, Naor and Lotspiech [25], which is one of the algorithms in the subset-cover framework. This technique is also used by Ghosh et al. [12] to obtain one-dimensional range queries in the three party setting. This work, on the other hand, is in the two party setting which is more challenging to realize.

For a full and complete binary tree of depth ℓ, $\mathcal{T}_\ell$, with nodes indexed by binary strings of length up to ℓ. Every node x of $\mathcal{T}_\ell$ defines a subset $\mathcal{S}_x$ of leaves, those in the full and complete subtree rooted at x. Conversely, for a given set of leaves $\mathcal{R}$, a directed Steiner Tree, denoted by $ST(\mathcal{R})$ in $\mathcal{T}_\ell$, is induced. $ST(\mathcal{R})$ is the minimal subtree (rooted at ϵ) of $\mathcal{T}_\ell$ that connects all the leaves in $\mathcal{R}$. Let $\mathcal{P} = \{p_1, \ldots, p_m\}$ be the set of nodes that are adjacent to nodes of outdegree one in $ST(\mathcal{R})$, which is the canonical covering of $[0, 2^\ell) \backslash \mathcal{R}$. Naor, Naor and Lotspiech [25] found that the size of $\mathcal{P}$ is upper-bounded by $|\mathcal{R}| \log(2^\ell / |\mathcal{R}|)$.

2.3 Background on Lattices

Lattices. Let n, m, and $q \geq 2$ be integers. For matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, define the m-dimensional lattice:

$$\Lambda^\perp(\mathbf{A}) = \big\{\mathbf{x} \in \mathbb{Z}^m : \ \mathbf{A} \cdot \mathbf{x} = \mathbf{0} \bmod q\big\} \subseteq \mathbb{Z}^m.$$

For any $\mathbf{u}$ in the image of $\mathbf{A}$, define $\Lambda^{\mathbf{u}}(\mathbf{A}) = \big\{\mathbf{x} \in \mathbb{Z}^m : \ \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \bmod q\big\}$.

Definition 1 **($\mathsf{SIS}_{n,m,q,\beta}$).** *Given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero vector $\mathbf{v} \in \Lambda^\perp(\mathbf{A})$ such that $\|\mathbf{v}\| \leq \beta$.*

If $m, \beta \in \mathsf{poly}(n)$ and $q > \beta \cdot \omega(\sqrt{n \log n})$, then the $\mathsf{SIS}_{n,m,q,\beta}$ problem is at least as hard as lattice problem SIVP_γ for some $\gamma = \beta \cdot \widetilde{\mathcal{O}}(\sqrt{n})$ (see, e.g., [10,23]).

Gaussian distributions. For integer $m > 0$, let $D_{\mathbb{Z}^m,\sigma}$ be the discrete Gaussian distribution over $\mathbb{Z}^m$ with parameter $\sigma > 0$. In the following lemmas, we review several well-known facts from [10].

Lemma 1. *We have* $\Pr\big[\, \|\mathbf{r}\| > \sigma\sqrt{m} \ \mid \ \mathbf{r} \hookleftarrow D_{\mathbb{Z}^m,\sigma} \big] \leq 2^{-m}$.

Lemma 2. *Let n be a positive integer, q be a prime, $m \geq 2n \log q$ and $\sigma = \Omega(\sqrt{n \log q \log n})$. Then, for a uniformly random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and for $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^m,\sigma}$, the distribution of $\mathbf{u} = \mathbf{A} \cdot \mathbf{r} \bmod q$ is statistically close to uniform over $\mathbb{Z}_q^n$. Moreover, the conditional distribution of $\mathbf{r}$ given $\mathbf{u}$ is $D_{\Lambda^{\mathbf{u}}(\mathbf{A}),\sigma}$.*

Lemma 3. *For $\sigma \geq \widetilde{\mathcal{O}}(\sqrt{m})$, the min-entropy of $D_{\mathbb{Z}^m,\sigma}$ is at least $m - 1$.*

When sampling a matrix $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_w] \in \mathbb{Z}^{m \times w}$, where $\mathbf{r}_i \hookleftarrow D_{\mathbb{Z}^m,\sigma}$ for all $i = 1, \ldots, w$, we will use the notation $\mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m \times w},\sigma}$.

Trapdoors for Lattices. We will employ the lattice trapdoors introduced by Micciancio and Peikert [22]. For any positive integer k, let $\mathbf{I}_k$ denote the identity matrix of order k. Let n be a positive integer, $q \in \mathsf{poly}(n)$ be a modulus and $w = n\lceil \log q \rceil$. Define the gadget matrix $\mathbf{G} = \mathbf{I}_n \otimes (1, 2, \ldots, 2^{\lceil \log q \rceil - 1}) \in \mathbb{Z}_q^{n \times w}$.

Let $m = \bar{m} + w$, for some $\bar{m} > w$. A trapdoor for matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is a matrix $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$ such that $\mathbf{A} \begin{bmatrix} \mathbf{R} \\ \mathbf{I}_w \end{bmatrix} = \mathbf{G}$. In particular, if $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}} \cdot \mathbf{R}]$, where $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$, then $\mathbf{R}$ is a trapdoor for $\mathbf{A}$.

Lemma 4 ([22]). *Let $n, q, w, \bar{m}, m$ be as above. Then, there exists a PPT algorithm* $\mathsf{TrapGen}(n, m, q)$ *that outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$, such that the distribution of $\mathbf{A}$ is statistically close to uniform.*

Moreover, for any $\mathbf{u} \in \mathbb{Z}_q^n$ and $\sigma = \Omega(\sqrt{n \log q \log n})$, there exists a PPT algorithm $\mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{u}, \sigma)$ *that outputs $\mathbf{r} \in \mathbb{Z}^m$ sampled from a distribution statistically close to $D_{\Lambda^{\mathbf{u}}(\mathbf{A}),\sigma}$.*

As shown by Micciancio and Peikert, a trapdoor for matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ can be efficiently extended into a trapdoor for any matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times (m+w)}$ of the form $\mathbf{B} = [\mathbf{A} \mid \mathbf{A}']$, where matrix $\mathbf{A}' \in \mathbb{Z}_q^{n \times w}$.

3 Zero-Knowledge Expressive Elementary Database from Trapdoor Mercurial Commitments

We construct a new flavor of size-hiding zero-knowledge database, called zero-knowledge expressive elementary database (ZK-EEDB). It allows databases D to be secretly committed in a public digest and several queries on D to be efficiently answered in zero-knowledge. The databases supported by ZK-EEDB are sets of records, which are key-value pairs $\big(x, \mathsf{D}(x)\big) \in [0, 2^\ell) \times [0, 2^\ell)$ and the queries supported by ZK-EEDB include queries over keys and values.

Besides membership over keys which was previously considered by Micali, Rabin and Kilian [21] in zero-knowledge elementary database, ZK-EEDB enables range queries over records of D, generalizing range queries over keys and values. We introduce the ability to generate proofs of correctness for answers to range queries over values in zero-knowledge with ZK-EEDB. The membership query over values, in this work, is the query which, given y, asks for the set $\mathsf{D}^{-1}(y) = \{x_i \mid x_i \in [\mathsf{D}] \text{ such that } \mathsf{D}(x_i) = y\}$. A range query over values is membership extended to a range of values $[a_y, b_y]$. From our techniques, we gain the ability to prove correctness of answers to range queries over records that is efficient for any super-polynomial range of keys.

First, we introduce new notations for values of a database D and the query types considered in ZK-EEDB. Following that, ZK-EEDB is formally defined and its security properties detailed. Then, we describe our techniques that enable efficient range queries over records of a database D. Finally, we end the section with a construction with TMC.

A Database of Values, D^{-1}. In this work, we consider queries over values of a database D in addition to queries over its keys. To achieve it efficiently, we use an alternate view of D, called D^{-1}, which is essentially a "reversed directory": namely, D^{-1} is the set $\{(y, \mathsf{D}^{-1}(y)) \mid y \in [0, 2^\ell)\}$, where $\mathsf{D}^{-1}(y) = \{x \mid x \in \mathsf{D} \text{ and } \mathsf{D}(x) = y\}$. The key-space of the database D^{-1} is thus the value-space of D and each key $y \in [\mathsf{D}^{-1}]$ has a value $\mathsf{D}^{-1}(y)$, which is the set of keys $x \in [\mathsf{D}]$ that are assigned the value y (i.e., $\mathsf{D}(x) = y$).

Queries in ZK-EEDB. Note that the answer to any query should come with a proof of correctness. We now describe a specific kind of query supported by our ZK-EEDB primitive which actually captures a total of six different queries.

- Range (Record) queries: Given a range $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y] \subset [0, 2^\ell) \times [0, 2^\ell)$, they return the set $\mathcal{L}$ of records such that $\mathcal{L} = \mathsf{D} \cap ([a_x, b_x] \times [a_y, b_y])$.
  - For general $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$, $[a_x, b_x]$ can be super-polynomial in ℓ.
  - Range queries over values (resp. keys) correspond to the input range $[0, 2^\ell) \times [a_y, b_y]$ (resp. $[a_x, b_x] \times [0, 2^\ell)$). For such queries, the interval $[a_y, b_y]$ (resp. $[a_x, b_x]$) can be super-polynomial or even exponential in ℓ.
  - Membership queries over records (resp. values and keys) correspond to the input range $[x, x] \times [y, y]$ (resp. $[0, 2^\ell) \times [y, y]$ and $[x, x] \times [0, 2^\ell)$).

3.1 Zero-Knowledge Expressive Elementary Database

ZK-EEDB has four algorithms: Init, ComDB, $\mathsf{ProveRQ}$, $\mathsf{VerifyRQ}$.

- $(crs, tk) \leftarrow \mathsf{Init}(1^\lambda)$: Takes as input security parameter λ and outputs a common reference string (CRS) crs and trapdoor key tk.
- $(com, \Delta) \leftarrow \mathsf{ComDB}(crs, \mathsf{D})$: Takes in crs and a database $\mathsf{D} = \{(x, \mathsf{D}(x))\}$. It returns a commitment com to D and a decommitment information Δ.
- $\Pi_{\mathfrak{R}} \leftarrow \mathsf{ProveRQ}(crs, (com, \Delta), \mathfrak{R})$: Inputs crs, a database commitment and decommitment information (com, Δ) and a range $\mathfrak{R}$. It returns a proof of correctness $\Pi_{\mathfrak{R}}$ of the range query with input range $\mathfrak{R} \subset [0, 2^\ell) \times [0, 2^\ell)$.
- $\mathcal{L} \leftarrow \mathsf{VerifyRQ}(crs, com, \mathfrak{R}, \Pi_{\mathfrak{R}})$: Inputs crs, a database commitment com, a range $\mathfrak{R} \subset [0, 2^\ell) \times [0, 2^\ell)$ and a purported proof $\Pi_{\mathfrak{R}}$. It returns

$$z = \begin{cases} \mathsf{D} \cap \mathfrak{R}, & \text{if the proof is correct;} \\ bad, & \text{if the proof is deemed invalid.} \end{cases}$$

We consider the same properties as in standard ZK-EDB protocols: namely, *completeness*, *soundness* and *zero-knowledge*, adapted to support the more expressive queries in ZK-EEDB. Correctness mandates that, for any query, correctly computed proofs satisfy the verification algorithm. Zero-knowledge requires that there exist an efficient simulator which is only granted oracle access to the database and outputs proofs for queries that are indistinguishable from those produced by a real prover using the real database as a witness. Soundness requires that no contradictory statements about the committed database can be proven by the adversary. Informally speaking, no PPT adversary can find two ranges $\mathfrak{R}, \mathfrak{R}'$ and proofs Π, Π' for which there exists a record $(x, y) \in \mathfrak{R} \cap \mathfrak{R}'$ that is in the answer to the first query but not the second. Formally, we have

- *Completeness*: For all databases D and all keys x, we have

  $$\begin{aligned}&\Pr[crs \leftarrow \mathsf{Init}(1^\lambda);\ (com, \Delta) \leftarrow \mathsf{ComDB}(crs, \mathsf{D});\\ &\quad \Pi_{\mathfrak{R}} \leftarrow \mathsf{ProveRQ}(crs, (com, \Delta), \mathfrak{R});\\ &\quad \mathsf{VerifyRQ}(crs, com, \mathfrak{R}, \Pi_{\mathfrak{R}}) \neq bad] = 1 - \nu(\lambda),\end{aligned}$$

  for some negligible function $\nu(\cdot)$.
- *Soundness*: For any PPT algorithm P', the probability

  $$\begin{aligned}&\Pr[crs \leftarrow \mathsf{Init}(1^\lambda);\ (com, \mathfrak{R}, \Pi, \mathfrak{R}', \Pi') \leftarrow \mathsf{P}'(crs);\\ &\quad \big(\mathsf{VerifyRQ}(crs, com, \mathfrak{R}, \Pi) = \mathcal{L} \neq bad\big)\\ &\ \wedge \big(\mathsf{VerifyRQ}(crs, com, \mathfrak{R}', \Pi') = \mathcal{L}' \neq bad\big)\\ &\ \wedge (\exists (x, y) \in \mathfrak{R} \cap \mathfrak{R}' \text{ s.t. } ((x, y) \in \mathcal{L}) \wedge ((x, y) \notin \mathcal{L}')],\end{aligned}$$

  is bounded by $\nu(\lambda)$, for some negligible function $\nu(\cdot)$.
- *Zero-Knowledge*: For any PPT adversary $\mathcal{A}$ and any efficiently computable database D, there exists an efficient simulator consisting of a triple of algorithms (SInit, SCom, $\mathsf{SProveQ}^{\mathsf{D}}$) such that the outputs of the following two experiment outputs are indistinguishable:

  Real experiment:
  1. Let $crs \leftarrow \mathsf{Init}(1^\lambda), (com, \Delta) \leftarrow \mathsf{ComDB}(crs, \mathsf{D})$ and $s_0 = \Pi_0 = \varepsilon$.
  2. For $1 \leq i \leq n$, we have $(\mathfrak{R}_i, s_i) \leftarrow \mathcal{A}(crs, com, \Pi_0, \ldots, \Pi_{i-1}, s_{i-1})$ and $\mathcal{A}$ gets a real proof $\Pi_i = \mathsf{ProveRQ}(crs, (com, \Delta), \mathfrak{R}_i)$.

  The experiment outputs $(crs, \mathfrak{R}_1, \Pi_1, \ldots, \mathfrak{R}_n, \Pi_n)$.

  Ideal experiment:
  1. Let $(crs', st_0) \leftarrow \mathsf{SInit}(1^\lambda), (com', st_1) \leftarrow \mathsf{SCom}(st_0)$ and $s_0 = \Pi'_0 = \varepsilon$.
  2. For $1 \leq i \leq n$, we have $(\mathfrak{R}_i, s_i) \leftarrow \mathcal{A}(crs', com', \Pi'_0, \ldots, \Pi'_{i-1}, s_{i-1})$ and $\mathcal{A}$ gets a simulated proof $\Pi'_i \leftarrow \mathsf{SProveRQ}^{\mathsf{D}}(crs', st_1, \mathfrak{R}_i)$.

  The experiment outputs $(crs', \mathfrak{R}_1, \Pi'_1, \ldots, \mathfrak{R}_n, \Pi'_n)$.

In the ideal experiment, $\mathsf{SProveQ}^{\mathsf{D}}$ is an oracle that is allowed to invoke a database oracle D and receive the set of records $\mathsf{D} \cap \mathfrak{R}$ for any range $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$ chosen by the adversary.

Here, a few comments about our security definitions are in order. We recall that, in size-hiding database commitments, the commitment *must* be shorter than the database since, otherwise, an upper bound on the database size is leaked. This naturally leads us to use statistically-hiding commitments, where we cannot properly speak of the "content" of a commitment since valid openings exist for any database. What matters is thus what the adversary is able to prove about the commitments it generates. In non-interactive size-hiding database commitments (at least under falsifiable assumptions), soundness can only be defined by preventing proofs for conflicting statements. In standard ZK-EDBs, it means one cannot prove distinct values for any key in a committed DB. For ZK-EEDBs, we extend it to range queries which are akin to batch queries. Our definition of soundness is thus adjusted to account for the answer being a set of records instead a value.

In the definitions of Ostrovsky *et al.* [28], soundness includes that, for any valid proofs produced by the prover, there exists a valid database compatible with the proven statements. This property is straightforward to show in our scheme and can be added to our model. Furthermore, although range queries can admit exponentially large ranges, it is still necessary to hide database size to maintain the zero-knowledge property of ZK-EEDB, which requires that verifiers leak nothing beyond the statements involved in all queries.

3.2 A Construction of ZK-EEDB from TMC: Initialization and Commitment Generation

In this section, we describe how to initialize a ZK-EEDB instance and commit to a database D by exploiting two size-hiding trees. Details of the proof generation and verification for ZK-EEDB queries are deferred to Section 4.

ZK-EEDB Database Commitments from TMC. To construct ZK-EEDBs, our idea is to use two Merkle trees to commit to the database D, each in a different representation. While ordinary ZK-EDBs consist of a single Merkle tree, ZK-EEDB relies on two size-hiding trees: (i) A (key) Merkle tree of height ℓ, which commits to a value $\mathsf{D}(x)$ at each leaf $x \in [\mathsf{D}]$; (ii) A (value) Merkle tree also of height ℓ that is two-tiered: each leaf y stores a commitment to the root value of a size-hiding Merkle tree that accumulates D_y^{-1}. Here, $\mathsf{D}_y^{-1} = \{(x, 1) \mid (x, y) \in \mathsf{D}\}$ is a zero-knowledge set encoded as a ZK-EDB with keys x and value 1 if and only if $(x, y) \in \mathsf{D}$.

The value Merkle tree can be seen as a commitment to the reversed database $\mathsf{D}_{com}^{-1} = \{(y, com_{\mathsf{D}_y^{-1}}) \mid \mathsf{D}^{-1}(y) \neq \emptyset\}$. Although defined differently earlier, we use D^{-1} from here on to denote D_{com}^{-1}. This is the main technique enabling efficient queries over values and records: Soft-opening paths in the value Merkle tree allow us to efficiently prove statements about non-membership of a value y (i.e., D contains no record of the form $(*, y)$). In existing single-tree-based constructions of ZK-EDBs, such queries are simply impossible to prove in zero-knowledge as each key must be separately proven to not have $\mathsf{D}(x) = y$ (which betrays the database size and is highly inefficient). However, in ZK-EEDB, the root value of the Merkle tree that accumulates D_y^{-1} is simply revealed to be empty by explaining the value stored at the leaf y in the value Merkle tree is a soft commitment C_y and showing a soft authentication path from y to the root.

With two commitments to the same database under different representations, we need to add checks to enforce that the two Merkle trees are consistent with each other. Whenever a record is proven to be in D via com_{D} and the first Merkle tree, the same record is also proven to be correctly committed in $com_{\mathsf{D}^{-1}}$ using the second Merkle tree. This prevents malicious provers from proving contradictory statements as both commitments have to agree at any record that has been proven to be in D. We insist that a cheating prover cannot win by using inconsistent databases in the two trees since, even by doing so, it will remain unable to prove contradictory statements without breaking the binding properties of the underlying mercurial commitment.

We now describe the initialization and commitment algorithms, Init and ComDB for ZK-EEDB from the TMC scheme, TMC.

- $(crs, tk) \leftarrow \mathsf{Init}(1^\lambda)$: compute and return $(crs, tk) \leftarrow \mathsf{TMC.Setup}(1^\lambda)$.
- $(com, \mathit{\Delta}) \leftarrow \mathsf{ComDB}(crs, \mathsf{D})$:
  1. Compute $(com_{\mathsf{D}}, \mathit{\Delta}_{\mathsf{D}}) \leftarrow \mathsf{BuildTree}(crs, \mathsf{D})$.
  2. For every y with $\mathsf{D}^{-1}(y) \neq \emptyset$, compute commitments of D_y^{-1} by running $(com_{\mathsf{D}_y^{-1}}, \mathit{\Delta}_{\mathsf{D}_y^{-1}}) \leftarrow \mathsf{BuildTree}(crs, \mathsf{D}_y^{-1})$.
  3. Compute $(com_{\mathsf{D}^{-1}}, \mathit{\Delta}_{\mathsf{D}^{-1}})$ with $\mathsf{BuildTree}(crs, \mathsf{D}^{-1})$.

  Return $(com = (com_{\mathsf{D}}, com_{\mathsf{D}^{-1}}), \mathit{\Delta} = (\mathit{\Delta}_{\mathsf{D}}, \{com_{\mathsf{D}_y^{-1}}\}_{y \in [\mathsf{D}^{-1}]}, \mathit{\Delta}_{\mathsf{D}^{-1}}))$.

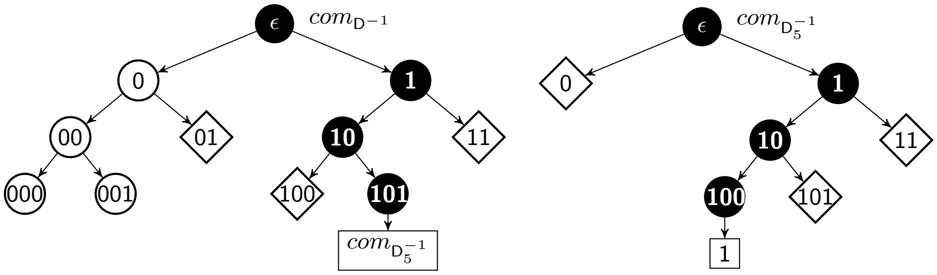

Fig. 1. The Value Merkle Tree in ZK-EEDB with Authentication Paths for $(4, 5) \in \mathsf{D}$.

4 Queries in ZK-EEDB

We first show how to prove correctness for answers to range queries in zero-knowledge for some database D committed with a Merkle tree and TMC scheme. Then, we apply the techniques to construct the $\mathsf{ProveRQ}$ and $\mathsf{VerifyRQ}$ algorithms in ZK-EEDB. Let the TMC scheme used, TMC, be implicit in the algorithms.

4.1 Range Queries with A Single Merkle Tree

For a single Merkle tree, a range query is an interval $[a, b] \subseteq [0, 2^\ell)$ of keys. Our range query proofs uses two key ideas: Steiner trees and a set of novel explanation algorithms. We can split the leaves in $[a, b]$ into two sets, $\mathcal{R} \subseteq [\mathsf{D}]$ with values in D, and the others, $[a, b] \backslash \mathcal{R}$. Proving correctness for $[a, b]$ means showing that every $x \in \mathcal{R}$ is a member of $[\mathsf{D}]$ and the remaining keys $[a, b] \backslash \mathcal{R}$ are not.

The Steiner tree characterizes the minimum set of nodes that have to be hard-opened to form the authentication paths for every leaf in $\mathcal{R}$. At the same time, it defines a polynomial-sized covering set for the remaining keys $[a, b] \backslash \mathcal{R}$. Then, the explanation algorithms are used to reveal that the covering set consists of soft commitments, so no hard authentication paths can be built from leaves in $[a, b] \backslash \mathcal{R}$ to the root of the Merkle tree.

Explanations for Trapdoor Mercurial Commitments. For our purposes, we introduce three new algorithms $\mathsf{Explain}, \mathsf{EVerify}, \mathsf{FakeExplain}$ to the syntax of TMC schemes. These algorithms reveal and verify that a commitment is a soft commitment and produce a "fake" proof that a fake commitment is a soft commitment. $\mathsf{Explain}$ is used by the prover when producing range proofs, $\mathsf{EVerify}$ is used by the verifier when checking if proofs are correct and $\mathsf{FakeExplain}$ is used by the simulator from the zero-knowledge property.

Note that Catalano *et al.* [2]'s construction of TMCs, and thus all known TMC schemes, can be easily adapted to support the three new algorithms introduced in this work. This is given in the full version of this work.

- $R \leftarrow \mathsf{Explain}(mpk; R)$: On input of the public commitment key mpk and random coin R such that $C = \mathbb{S}\mathsf{Commit}(mpk; R)$, it outputs the random coin R that explains C as a soft commitment.
- $\mathsf{EVerify}(mpk, C, R)$: On input of the public commitment key mpk, a commitment C and random coins R, it accepts if R is deemed as convincing evidence that C is a soft commitment.
- $R' \leftarrow \mathsf{FakeExplain}(msk; R)$: On input of the public commitment key mpk and random coins R such that $C = \mathsf{MFake}(mpk; R)$, this algorithm outputs random coins R' such that $C = \mathbb{S}\mathsf{Commit}(mpk; R')$.

It is straightforward to see that $\mathsf{EVerify}$ will only accept if the inputs are soft commitment, explanation or fake commitment, fake explanation pairs. If an adversary can produce explanations for some hard commitment that $\mathsf{EVerify}$ accepts, then mercurial binding is broken: The explanation can be adapted to produce soft-openings to any message like fake commitments which contradicts the mercurial binding property that hard commitments can only be hard-/soft-opened to a unique message.

With these three new algorithms, we require an additional equivocation property, *soft-explain* (SE) equivocation, for the security of TMC schemes.

- *SE Equivocation*: The real game provides $\mathcal{A}$ with a soft commitment $C = \mathbb{S}\mathsf{Commit}(mpk; R)$ and the corresponding random coins R. The ideal game provides $\mathcal{A}$ with a fake commitment $C = \mathsf{MFake}(msk; R)$ and a fake explanation $R' \leftarrow \mathsf{FakeExplain}(msk; R)$ of C as a soft commitment.

Optimized Proof of Membership for an Interval. A naive method to prove membership of a set of keys of D lying in some interval $[a, b]$, $\mathcal{R}$, is to return $|\mathcal{R}|$ many hard authentication paths. This is sub-optimal as there are duplicated hard openings as authentication paths merge closer to the root of the Merkle tree. We show how the Steiner tree yields the optimal set of hard openings.

For a set of leaves $\mathcal{R} \subseteq [\mathsf{D}] \cap [a, b]$, let $ST(\mathcal{R})$ be the Steiner tree of $\mathcal{R}$, the minimal subtree connecting the leaves of $\mathcal{R}$ to the root. We use $ST(\mathcal{R})_j$ to mean the set of nodes in $ST(\mathcal{R})$ at depth j. We define the authentication Steiner tree of $\mathcal{R}$, $\Pi_{\mathcal{R}}$, as the set of hard openings for each node in $ST(\mathcal{R})$: namely, hard openings π_x to $\mathsf{D}(x)$ for each leaf $x \in \mathcal{R}$, and π_y to $(C_y, C_{y'})$ for each internal node $y \in ST(\mathcal{R})\backslash\mathcal{R}$. By definition, hard authentication paths for all $x \in \mathcal{R}$ are in $\Pi_{\mathcal{R}}$ above which has $\mathcal{O}(|\mathcal{R}| \cdot \ell)$ nodes. The mechanism and its verification is formalized in $\mathbb{H}\mathsf{OpenST}$ and $\mathsf{VerifyST}$ with $(com, \Delta) \leftarrow \mathsf{BuildTree}(crs, \mathsf{D})$.

- $\Pi_{\mathcal{R}} \leftarrow \mathbb{H}\mathsf{OpenST}(crs, (com, \Delta), \mathcal{R})$: With inputs CRS crs, database commitment com, decommitment information Δ and set $\mathcal{R} \subseteq [\mathsf{D}]$, return $\Pi_{\mathcal{R}}$ as follows:
  1. For each leaf $x \in \mathcal{R}$, compute $\pi_x \leftarrow \mathbb{H}\mathsf{Open}(crs, \mathsf{D}(x); R_x)$.
  2. For j from $\ell - 1$ to 0 and $z \in ST(\mathcal{R})_j$, the set of nodes in $ST(\mathcal{R})$ at depth j, compute $\pi_z \leftarrow \mathbb{H}\mathsf{Open}(crs, (C_{z0}, C_{z1}); R_z)$.

  Set $\Pi_{\mathcal{R}} = (\{(x, \mathsf{D}(x)), \pi_x\}_{x\in\mathcal{R}}, \{C_z, C_{z'}, \pi_z\}_{z\in ST(\mathcal{R})\backslash\mathcal{R}})$.
- $\mathcal{L} \leftarrow \mathsf{VerifyST}(crs, com, \mathcal{R}, \Pi_{\mathcal{R}})$: With inputs CRS crs, database commitment com, set $\mathcal{R}$,
  1. For each leaf $x \in \mathcal{R}$, compute $\mathbb{H}\mathsf{Verify}(crs, \mathsf{D}(x), C_x, \pi_x)$ and continue if all verifications accept. Otherwise, set and output $\mathcal{L} = bad$.
  2. For j from $\ell - 1$ to 0 and $z \in ST(\mathcal{R})_j$, the set of nodes in $ST(\mathcal{R})$ at depth j, compute $\mathbb{H}\mathsf{Verify}(crs, (C_{z0}, C_{z1}), C_z, \pi_z)$ and continue if all verifications accept. Otherwise, set and output $\mathcal{L} = bad$.

  Return $\mathcal{L} = \mathcal{R}$ if $\mathcal{L} \neq bad$.

Proof of Non-Membership for an Interval. With the authentication Steiner tree for $\mathcal{R}$ proving that keys $x \in \mathcal{R}$ are in $[\mathsf{D}]$, we need to show that the other keys in $[a, b]\backslash\mathcal{R}$ do not appear in D. This is achieved with a crucial observation: If an internal node y' is a soft commitment or nil, then no descendant of y' has a valid hard authentication path and cannot be in $[\mathsf{D}]$. So, we use $\mathsf{Explain}$ to prove that no leaves in $[a, b]\backslash\mathcal{R}$ have a hard authentication path to the root. In particular, if $\mathcal{P}$ is the canonical covering of $[a, b]\backslash\mathcal{R}$, we have $C_x = nil$ or $\mathbb{S}\mathsf{Commit}(mpk; R_x)$ for any node $x \in \mathcal{P}$ after $\mathsf{BuildTree}(crs, \mathsf{D})$.

The values C_z of $z \in \mathcal{P}$ are explained as soft commitments to show that the leaves $[a, b]\backslash\mathcal{R}$ cannnot be involved in a proof of membership. With $\mathcal{R} = \{x_1, \ldots, x_m\}$. the canonical coverings of intervals $[a, x_1 - 1]$ and $[x_m + 1, b]$ may not be siblings of nodes in $ST(\mathcal{R})$ but descendants instead. In those cases, we compute soft authentication paths from these canonical coverings to the ancestors which are siblings of some nodes in $ST(\mathcal{R})$. Those $z \in \mathcal{P}$ whose siblings z' are in $ST(\mathcal{R})$ do not need additional proof elements. The entire process adds $\mathcal{O}\big(|\mathcal{R}| \cdot \log((b - a)/|\mathcal{R}|)\big)$ nodes which is only a constant factor larger than $|ST(\mathcal{R})| = |\mathcal{R}| \cdot \ell$. Thus, the entire range proof has size $\mathcal{O}(|\mathcal{R}| \cdot \ell)$, independent of the length of the input interval and allows for exponentially large inputs.

- $\Pi_{[a,b]} \leftarrow \mathsf{OpenI}(crs, (com, \Delta), [a, b])$: If $a = b$, prove membership of a with $\mathbb{H}\mathsf{OpenPath}$ and non-membership with $\mathbb{S}\mathsf{OpenPath}$. Otherwise proceed as follows. Let $\mathcal{R}$ be the set of keys in $[\mathsf{D}] \cap [a, b]$.

- If $\mathcal{R} = \emptyset$, set $\mathit{\Pi}_{[a,b],\mathcal{R}} = \mathit{nil}$ and let $\mathcal{P}$ be the canonical covering of the leaves in $[a,b]$. For each $x \in \mathcal{P}$:
  1. Compute $C_q \leftarrow \mathbb{S}\mathsf{Commit}(\mathit{crs}; R_q)$ if $C_q = \mathit{nil}$ for $q = x, x'$.
  2. Compute $R_x \leftarrow \mathsf{Explain}(\mathit{crs}, C_x; R_x)$.
  3. For i from $|x| - 1$ to 0, compute $C_q \leftarrow \mathbb{S}\mathsf{Commit}(\mathit{crs}; R_{x|_i})$ if either of $q = x|_i, (x|_i)'$ has not been computed previously. Then, compute $\tau_{x|_i} \leftarrow \mathbb{S}\mathsf{Open}(\mathit{crs}, (C_{x|_i0}, C_{x|_i1}); R_{x|_i})$.

  Set $\mathit{\Pi}_{[a,b],\mathcal{P}} = \{R_x, \{C_{x|_i0}, C_{x|_i1}, \tau_{x|_i}\}_{0 \le i \le |x|-1}\}_{x \in \mathcal{P}}$, proofs that C_x is a soft commitment and committed to *com* for $x \in \mathcal{P}$.
- Otherwise, $\mathcal{R} \neq \emptyset$ and compute the authentication Steiner tree of $\mathcal{R}$, $\mathit{\Pi}_{[a,b],\mathcal{R}} \leftarrow \mathbb{H}\mathsf{OpenST}(\mathit{crs}, (\mathit{com}, \mathit{\Delta}), \mathcal{R})$.
  1. (*Explain canonical covering.*) Let $\mathcal{P}$ be the canonical covering of the keys in $[a,b]\backslash\mathcal{R}$, for $x \in \mathcal{P}$, compute $R_x \leftarrow \mathsf{Explain}(\mathit{crs}, C_x; R_x)$
  2. (*Prove connection to $ST(\mathcal{R})$.*) If $x' \notin ST(\mathcal{R})$, let h_x be such that $x|_{h_x} = y'$ for some $y \in ST(\mathcal{R})$. For i from $|x| - 1$ to h_x, compute $\tau_{x|_i} \leftarrow \mathbb{S}\mathsf{Open}(\mathit{crs}, (C_{x|_i0}, C_{x|_i1}); R_{x|_i})$.

  Set $\mathit{\Pi}_{[a,b],\mathcal{P}} = \{R_x, \{C_{x|_i0}, C_{x|_i1}, \tau_{x|_i}\}_{h_x \le i \le |x|-1}\}_{x \in \mathcal{P}}$, proving C_x are soft commitments and their paths to *com* meet $ST(\mathcal{R})$ for $x \in \mathcal{P}_{[a,b]}$.

Output $\mathit{\Pi}_{[a,b]} = (\mathit{\Pi}_{[a,b],\mathcal{P}}, \mathit{\Pi}_{[a,b],\mathcal{R}})$ and add the randomness of any commitments computed to $\mathit{\Delta}$.

– $\mathcal{L} \leftarrow \mathsf{VerifyI}(\mathit{crs}, \mathit{com}, \mathfrak{R}, \mathit{\Pi}_{[a,b]})$: If $a = b$, $\mathit{\Pi}_{[a,b]} = \mathit{\Pi}_a$ and compute $y \leftarrow \mathsf{VerifyPath}(\mathit{crs}, \mathit{com}, a, \mathit{\Pi}_a)$. Set $\mathcal{L}' = \{(a, y)\}$ if $y \notin \{\perp, \mathit{bad}\}$ and $\mathcal{L}' = \emptyset$ if $y = \perp$. If $a \neq b$, let the proof $\mathit{\Pi}_{[a,b]} = (\mathit{\Pi}_{[a,b],\mathcal{P}}, \mathit{\Pi}_{[a,b],\mathcal{R}})$, where $\mathcal{R}$ denotes the set of keys returned. Set $\mathcal{L}' = \emptyset$ and proceed as follows.

- If $\mathcal{R} = \emptyset$, $\mathit{\Pi}_{[a,b],\mathcal{R}} = \mathit{nil}$, $\mathit{\Pi}_{[a,b],\mathcal{P}} = \{R_x, \{C_{x|_i0}, C_{x|_i1}, \tau_{x|_i}\}_{0 \le i \le |x|-1}\}_{x \in \mathcal{P}}$. For each $x \in \mathcal{P}$, where $\mathcal{P}$ is the canonical covering of $[a,b]$:

  a. Compute $\mathsf{EVerify}(\mathit{crs}, C_x, R_x)$ to check that C_x is a soft commitment. Set $y = \mathit{bad}$ if it is not.

  b. For i from $|x| - 1$ to 0, compute $\mathbb{S}\mathsf{Verify}(\mathit{crs}, (C_{x|_i0}, C_{x|_i1}), C_x, \tau_x)$ and set $y = \mathit{bad}$ if any verification fails.
- Otherwise, $(\mathit{\Pi}_{[a,b],\mathcal{R}} = (\{(x, \mathsf{D}(x)), \pi_x\}_{x \in \mathcal{R}}, \{C_z, C_{z'}, \pi_z\}_{z \in ST(\mathcal{R})\backslash\mathcal{R}})$ and $\mathit{\Pi}_{[a,b],\mathcal{P}} = \{R_x, \{C_{x|_i0}, C_{x|_i1}, \tau_{x|_i}\}_{h_x \le i \le |x|-1}\}_{x \in \mathcal{P}}$.
  1. Compute $\mathcal{L}' \leftarrow \mathsf{VerifyST}(\mathit{crs}, \mathit{com}, \mathcal{R}, \mathit{\Pi}_{[a.b],\mathcal{R}})$ to check the authentication Steiner tree.
  2. (*Check canonical covering of* $[a,b]\backslash\mathcal{R}$.) Let $\mathcal{P}$ be the canonical covering of the keys in $[a,b]\backslash\mathcal{R}$. For each $x \in \mathcal{P}$:

     a. Compute $\mathsf{EVerify}(\mathit{crs}, C_x, R_x)$ and set $y = \mathit{bad}$ if it fails.

     b. For i from $|x| - 1$ to h_x, compute $\mathbb{S}\mathsf{Verify}(\mathit{crs}, (C_{x0}, C_{x1}), \tau_x)$ and set $y = \mathit{bad}$ if any verification fails.

If $\mathcal{L}'$ and $y \neq \mathit{bad}$, set and output $\mathcal{L} = \mathcal{L}'$.

4.2 Range Queries over Records in ZK-EEDB

Let $(crs, tk) \leftarrow \mathsf{Init}(1^\lambda)$ and $(com, \varDelta) \leftarrow \mathsf{ComDB}(crs, \mathsf{D})$ be the ZK-EEDB commitment and decommitment information of a database D and consider an arbitrary range $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$. Correctness of its answer is proved in zero-knowledge with several membership and range proofs in the Merkle trees built in ComDB. Due to space constraints, we sketch the algorithms for range proof generation and verification and defer its formal description the full version of this work.

$\mathsf{ProveRQ}$**.** The value Merkle tree can be seen as a two-tiered size-hiding commitment to D, by storing commitments to $\mathsf{D}^{-1}(y)$ for every possible value in the universe $y \in [0, 2^\ell)$. Proofs of membership of a record (x, y) on the value Merkle tree would comprise of two parts. First, the committer proves that the value at some leaf x is a hard commitment to 1 in the commitment $com_{\mathsf{D}_y^{-1}}$. This shows that the record (x, y) is in some database committed in some commitment $com_{\mathsf{D}_y^{-1}}$, which we next prove to be the commitment to D_y^{-1}. We achieve this by proving that $com_{\mathsf{D}_y^{-1}}$ is committed in the value at leaf y in the ZK-EEDB commitment of the value Merkle tree, $com_{\mathsf{D}^{-1}}$.

Moving into the sketch of the algorithm, we begin with the most straightforward case: range queries over keys with $\mathfrak{R} = [a_x, b_x] \times [0, 2^\ell)$. For this, we simply use OpenI to prove (non-)membership of all keys in $[a_x, b_x]$ of the key Merkle tree. Then, for consistency, we prove that each record $(x, \mathsf{D}(x))$ is committed in the value Merkle tree as well. Next, for range queries over values with range $[0, 2^\ell) \times [a_y, b_y]$, the procedure is similar. We use OpenI on the first tier of the value Merkle tree with the interval $[a_y, b_y]$, which proves that some values do not occur in D and the remaining values store commitments to some non-empty D_y^{-1}. After that, we simply reveal the entire Merkle tree for each non-empty D_y^{-1} with OpenI on the interval $[0, 2^\ell)$. Finally, for consistency, we generate the hard authentication path from leaf x to the root of the key Merkle tree for each record (x, y) that is shown to be in D from the value Merkle tree.

Finally, we describe the proof generation for range queries over records with $\mathfrak{R} = [a_x, b_y] \times [a_y, b_y]$. We start in the first tier of the value Merkle tree, and prove that $com_{\mathsf{D}_y^{-1}}$ is the commitment to D_y^{-1} for each $y \in [a_y, b_y]$; a hard (resp. soft) authentication path from y to the root of the value Merkle tree is generated for those that are non-empty (resp. empty) in $[a_x, b_x]$. Then, for each $y \in \mathsf{D}_y^{-1}$, we use OpenI to prove (non-)membership of all the keys in the interval $[a_x, b_x]$. Consistency is proven in the same way as range queries over values.

$\mathsf{VerifyRQ}$**.** To verify range proofs, we verify proofs for the key and value Merkle tree separately for the set of records $\mathcal{L}$ returned. For the key Merkle tree, the process is straightforward; we either verify a (non-)membership proof for an interval $[a_x, b_x]$ in range queries over keys or a set of hard authentication paths in the other two range queries. Proofs for the value Merkle tree consists of verifying that records are committed in some commitments which purport to be of D_y^{-1}. These supposed commitments of D_y^{-1} are then verified to be what they are by checking that they are committed in $com_{\mathsf{D}^{-1}}$. Proofs fail verification if any of the sub-proofs are incorrect.

For range queries over keys, honestly computed range proofs Π contain a (non-)membership proof for all keys $x \in [a_x, b_x]$ which we verify with $\mathsf{VerifyI}$. Then, for consistency, Π also contains individual hard authentication paths for each record $(x, \mathsf{D}(x)) \in \mathcal{L}$ from leaf x to a claimed commitment of $\mathsf{D}^{-1}(\mathsf{D}(x))$ and hard authentication path from leaf $\mathsf{D}(x)$ to the root of the value Merkle tree whose value is $com_{\mathsf{D}^{-1}}$. These are verified with the authentication path verification algorithm $\mathsf{VerifyPath}$

Let $\mathcal{L}$ denote the set of records returned by the prover and Π be the range proof. For range queries over values, the set $\mathcal{L}$ can be partitioned into $\mathcal{L} = \bigcup_{y \in \mathcal{V}} \{(x_i, \mathsf{D}(x_i) = y)\}$, where $\mathcal{V}$ is the set of unique values that occur in $\mathcal{L}$. Then, we only need to verify (non-)membership proofs for all keys in $\mathsf{D}^{-1}(y)$ for $y \in \mathcal{V}$, to check that the records returned are correctly commited in some claimed commitment of $\mathsf{D}^{-1}(y)$. Finally, we verify (non-)membership proofs for the interval $y \in [a_y, b_y]$ of the value Merkle tree using $\mathsf{VerifyI}$ to check that the claimed commitments to D_y^{-1} for $y \in [a_y, b_y]$ are valid and the remaining D_y^{-1}'s are empty. Consistency checks are straightforward, each record returned is checked to have a valid hard authentication path in Π with $\mathsf{VerifyPath}$.

Lastly, for range queries over records, consistency checks are identical to range queries over values and so we focus on the differences in the value Merkle tree. Instead of checking only $y \in \mathcal{V}$, we have to do verify the (non-)membership proof for the interval $[a_x, b_x]$ with the claimed $com_{\mathsf{D}_y^{-1}}$ for every $y \in [a_y, b_y]$. This is done with $\mathsf{VerifyI}$. Finally, we check that the claimed commitments to D_y^{-1} are correctly committed with valid hard or soft authentication paths, for every $y \in [a_y, b_y]$, to the root of the value Merkle tree whose value is $com_{\mathsf{D}^{-1}}$.

Proof Sizes. There are three cases with different input ranges $\mathfrak{R}$ and proof sizes, which is taken to be the number of nodes to open or explain. Let $\mathcal{L}$ denote the answer to the range query with input $\mathfrak{R} = [a_x, b_x] \times [a_y, b_y]$.

First, the general case where $[a_x, b_x]$ and $[a_y, b_y]$ are not $[0, 2^\ell)$. We partition $\mathcal{L} = \bigcup_{y \in [a_y, b_y]} \mathcal{L}_y$ based on the value of the record. The proof consists of $(b_y - a_y)$ authentication paths in the value Merkle tree, the same number of authentication Steiner trees, one in every Merkle tree with root value $com_{\mathsf{D}_y^{-1}}$ for $y \in [a_y, b_y]$ and finally $|\mathcal{L}|$ authentication paths in the key Merkle tree. The Steiner trees and paths would have $\mathcal{O}(|\mathcal{L}_y|\ell)$ and ℓ nodes each respectively. This brings the total proof size to $\mathcal{O}(\big((b_y - a_y)(1 + K) + |\mathcal{L}|\big)\ell)$ nodes, where $K = \max_{y \in [a_y, b_y]} |\mathcal{L}_y|$.

Next, we consider range queries over values with $\mathfrak{R} = [0, 2^\ell) \times [a_y, b_y]$. Let $\mathcal{V}$ be the set of distinct values in the answer set $\mathcal{L}$, which we partition into disjoint subsets based on the value of the record, i.e., $\mathcal{L} = \bigcup_{y \in \mathcal{V}} \mathcal{L}_y$. Since the only difference between this and the general case is that use OpenI, we have only one authentication Steiner tree for the value Merkle tree and $|\mathcal{V}|$ many Steiner trees for each D_y^{-1} with $y \in \mathcal{V}$. Therefore, the proof size for this case is $\mathcal{O}(\big(|\mathcal{L}| + |\mathcal{V}|(1 + K)\big)\ell)$ with $K = \max_{y \in \mathcal{V}} |\mathcal{L}_y|$.

Lastly, for range queries over keys with $\mathfrak{R} = [a_x, b_x] \times [0, 2^\ell)$, the proof consists of the authentication Steiner tree of $\mathcal{L}$ in the key Merkle tree and $\mathcal{O}(|\mathcal{L}|)$ authentication paths in the Merkle tree and some set of Merkle trees with root value $com_{\mathsf{D}_y^{-1}}$ where $(x, y) \in \mathcal{L}$. In total, the proof size is $\mathcal{O}(|\mathcal{L}|\ell)$.

Overall, $\mathsf{ProveRQ}$ supports super-polynomial intervals $[a_x, b_x]$ over the key-space for any query and value-space for range queries over values. For range queries over records, only polynomial length intervals $[a_y, b_y]$ are supported.

4.3 Security of the ZK-EEDB Construction

Recall that ZK-EEDB has three properties, correctness, soundness and zero-knowledge. Correctness can be verified from the construction of ZK-EEDB easily.

Theorem 1. *The ZK-EEDB resulting from this construction is sound if the TMC scheme is mercurial-binding.*

Proof. Suppose that the adversary can produce two contradicting range queries with valid proofs, $\mathfrak{R}, \Pi$ and $\mathfrak{R}', \Pi'$. There must exist a record $(x, y) \in \mathfrak{R} \cap \mathfrak{R}'$ in $\mathcal{L} = \mathsf{VerifyRQ}(crs, com, \mathfrak{R}, \Pi)$ but not $\mathcal{L}' = \mathsf{VerifyRQ}(crs, com, \mathfrak{R}', \Pi')$. There are two cases in this situation: (i) There exists another record $(x, y') \in \mathcal{L}'$ such that $y' \neq y$; (ii) There exists no $y' \in [0, 2^\ell)$ such that $(x, y') \in \mathcal{L}'$.

In case (i), both (x, y) and (x, y') have valid proofs that they are committed in com_{D}. This means that Π and Π' contain two valid hard decommitments to distinct values in two distinct authentication paths for the leaf x of the Merkle tree of com_{D}. This breaks the mecurial-binding property of the TMC scheme in the same way as in the proof of the soundness property in ordinary ZK-EDBs.

In case (ii), there exists no record with key x in $\mathcal{L}'$. This implies that Π' contains a proof that $(x, 1) \notin \mathsf{D}_y^{-1}$ and therefore $x \notin \mathsf{D}^{-1}(y)$. However, Π does contain a proof that $(x, 1) \in \mathsf{D}_y^{-1}$, leading to a contradiction between Π and Π'.

For this to happen, the first possibility is that the two proofs differ in their commitments $com_{\mathsf{D}_y^{-1}}$ of $\mathsf{D}^{-1}(y)$. If so, the value at leaf y of the Merkle tree of $com_{\mathsf{D}^{-1}}$ has a valid hard opening to one value in Π while in Π', the value at the leaf or some node along its path to the root is either explained as a soft commitment or soft-opened to a message contradicting the hard opening. This contradicts the mercurial-binding property of the TMC scheme, which says that a mercurial commitment cannot be soft-opened to one message and hard-opened to a different one. The second possibility is that the commitments $com_{\mathsf{D}_y^{-1}}$ are identical in both Π and Π' but the two proofs depart within the Merkle tree with root value $com_{\mathsf{D}_y^{-1}}$. Since Π proves that $(x, 1) \in \mathsf{D}_y^{-1}$, it contains a hard authentication path from leaf x to the root. However, Π' proves that $(x, 1) \notin \mathsf{D}_y^{-1}$, meaning either: (a) The value at leaf x is a soft commitment; (b) Some node along the path from leaf x to the root is explained as a soft commitment in Π'. Either way, Π shows that the value at some node is a hard commitment whereas Π' shows that otherwise, the value at the same node either explained as a soft commitment or soft-opened to a message that contradicts the hard opening in Π. As before, this contradicts the mercurial-binding property of the TMC scheme.

Theorem 2. *The ZK-EEDB resulting from this construction satisfies the zero-knowledge property if the TMC scheme satisfies the four equivocation properties.*

Proof. The zero-knowledge property follows from the equivocation properties enabled by the trapdoor in the TMC scheme. The ZK-EEDB simulator (SInit, SCom, $\mathsf{SProveRQ}^{\mathsf{D}}$), which is constructed below, is similar to the ZK-EDB simulator. However, a key change is that $\mathsf{SProveRQ}^{\mathsf{D}}$ additionally uses $\mathsf{FakeExplain}$. To simulate range proofs, $\mathsf{FakeExplain}$ allows explaining fake commitments as soft commitments when some subtrees have to be proved empty.

- $(crs', st_0) \leftarrow \mathsf{SInit}(1^\lambda)$: Run $(crs, tk) \leftarrow \mathsf{Init}(1^\lambda)$ and output the common reference string and simulator state $(crs' = crs, st_0 = tk)$.
- $(com', st_1) \leftarrow \mathsf{SCom}(st_0)$: Compute $com'_{\mathsf{D}} \leftarrow \mathsf{MFake}(crs'; R_0)$ and $com'_{\mathsf{D}^{-1}} \leftarrow \mathsf{MFake}(crs'; R_1)$ and output $(com' = (com'_{\mathsf{D}}, com'_{\mathsf{D}^{-1}}), st_1 = (R, tk))$.
- $\Pi' \leftarrow \mathsf{SProveRQ}^{\mathsf{D}}(crs', st_1, \mathfrak{R} = [a_x, b_x] \times [a_y, b_y])$: Obtain the set $\mathcal{S} = \mathsf{D} \cap \mathfrak{R}$ by querying the database oracle and let $\mathcal{S}' \in st_1$ contain the commitments and proofs that were computed in previous queries. We denote with $\mathcal{S}_{\mathsf{D}^{-1}}$, the set of distinct values in $\mathcal{S}$. Then, let $[\mathcal{S}]_{\mathsf{D}^{-1}_y}$ be the set of keys in $[\mathcal{S}]$ whose values in D are y. Compute Π' as follows:
  1. The answer defines several Steiner trees and paths that are needed to prove the correctness of the answer to the adversary.
     a. If $[a_y, b_y] = [0, 2^\ell)$, then $\mathcal{S}$ induces an authentication Steiner tree, $ST([\mathcal{S}])$ in the Merkle tree of com'_{D} and $|\mathcal{S}|$ many authentication paths, $\mathcal{L}$ and $\mathcal{L}_y$, in the Merkle trees of $com'_{\mathsf{D}^{-1}}$ and $com'_{\mathsf{D}^{-1}_y}$ for $y \in \mathcal{S}_{\mathsf{D}^{-1}}$ respectively.
     b. If $[a_x, b_x] = [0, 2^\ell)$, then $\mathcal{S}_{\mathsf{D}^{-1}}$ defines an authentication Steiner tree, $ST(\mathcal{S}_{\mathsf{D}^{-1}})$ in the Merkle tree of $com'_{\mathsf{D}^{-1}}$ and similar Steiner trees $ST([\mathcal{S}]_y)$ in the Merkle trees of $com'_{\mathsf{D}^{-1}_y}$ for $y \in \mathcal{S}_{\mathsf{D}^{-1}}$. Finally, $[\mathcal{S}]$ defines $|[\mathcal{S}]|$ many authentication paths $\mathcal{L}$ in the Merkle tree of com'_{D}.
     c. If neither $[a_x, b_x]$ not $[a_y, b_y]$ are $[0, 2^\ell)$, then $\mathcal{S}_{\mathsf{D}^{-1}}$ defines $|ST(\mathcal{S}_{\mathsf{D}^{-1}})|$ paths, $\mathcal{L}$ in the Merkle tree of $com'_{\mathsf{D}^{-1}}$ and Steiner trees $ST([\mathcal{S}]_y)$ in the Merkle trees of $com'_{\mathsf{D}^{-1}_y}$ for $y \in \mathcal{S}_{\mathsf{D}^{-1}}$. $[\mathcal{S}]$ also defines an authentication Steiner tree $ST([\mathcal{S}])$ in the Merkle tree of com'_{D}.
  2. For each range type, let $\mathcal{N}$ be the set of nodes in the trees and paths induced by the answer $\mathsf{D} \cap \mathfrak{R}$. Then, for every node $x \in \mathcal{N} \backslash \mathcal{S}'$, compute fake commitments $C_x \leftarrow \mathsf{MFake}(crs; R_x)$.
  3. For the fake commitments created in Step 2 and their parents, compute appropriate hard and soft decommitments and explanations using $\mathbb{H}\mathsf{Equivocate}$, $\mathbb{S}\mathsf{Equivocate}$ and‘’ $\mathsf{FakeExplain}$ to simulate an honest proof.
  4. Add the fake commitments, hard and soft decommitments and explanations computed in Steps 1 and 2 to the state st_1.

The output of the simulator is indistinguishable from that of an honest prover because of the equivocation properties of the TMC scheme used. There are two types of outputs from the simulator, the CRS from initialization and fake commitments, decommitments and explanations in proofs to queries from the adversary. The simulated CRS is indistinguishable from a real CRS one as both are trapdoor mercurial commitment keys. From the four equivocation properties of the TMC scheme, the joint distribution of fake commitments and their hard/soft equivocations or explanations are statistically indistinguishable from hard/soft commitments and their hard/soft openings or explanations. □

5 Lattice Instantiations

5.1 A Trapdoor Mercurial Commitment from Standard Lattices

Let $\lambda \in \mathbb{N}$ be a security parameter. The scheme works with message space $\mathcal{M} = \{0,1\}^l$, where $l \in \mathsf{poly}(\lambda)$. For a dimension $n = \mathcal{O}(\lambda)$ and prime modulus $q = \widetilde{\mathcal{O}}(l \cdot n^2 + n^4)$, let $w = n\lceil \log q \rceil$, $\bar{m} = 2n\lceil \log q \rceil$ and $m = \bar{m} + w$. Choose a Gaussian parameter $\sigma = \Omega(\sqrt{n \log q \log n})$.

- $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$: Choose a matrix $\mathbf{A}_0 \hookleftarrow U(\mathbb{Z}_q^{n \times l})$. Run algorithm $\mathsf{TrapGen}(n, m, q)$ (Lemma 4) to generate a pair $(\mathbf{A}_1, \mathbf{T})$, where $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$ is statistically close to uniform and $\mathbf{T} \in \mathbb{Z}^{\bar{m} \times w}$ is its trapdoor.
  Output $mpk = (\mathbf{A}_0, \mathbf{A}_1)$ and $msk = \mathbf{T}$.
- $\mathbf{C} \leftarrow \mathbb{H}\mathsf{Commit}(mpk, \boldsymbol{\mu}; (\mathbf{R}, \mathbf{r}))$: Given a message $\boldsymbol{\mu} \in \{0,1\}^l$ and randomness $\mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m \times w}, \sigma}$ and $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^{m+w}, \sigma}$, define $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1] \in \mathbb{Z}_q^{n \times (m+w)}$, where $\mathbf{B}_1 = \mathbf{A}_1 \cdot \mathbf{R} \in \mathbb{Z}_q^{n \times w}$. Then, compute $\mathbf{c} = \mathbf{A}_0 \cdot \boldsymbol{\mu} + \mathbf{B} \cdot \mathbf{r} \in \mathbb{Z}_q^n$ and output the hard commitment $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times w}$.
- $\pi \leftarrow \mathbb{H}\mathsf{Open}(mpk, \mu; (\mathbf{R}, \mathbf{r})$: Output $\pi = (\mathbf{R}, \mathbf{r}) \in \mathbb{Z}^{m \times w} \times \mathbb{Z}^{m+w}$.
- $\mathbb{H}\mathsf{Verify}(mpk, \boldsymbol{\mu}, \mathbf{C}, \pi)$: Given a commitment $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times w}$ and a purported hard opening $\pi = (\mathbf{R}, \mathbf{r})$, proceed as follows.
  1. Return 0 if $\mathbf{R} = [\mathbf{r}_1 \mid \ldots \mid \mathbf{r}_w]$ has a column such that $\|\mathbf{r}_i\| > \sigma\sqrt{m}$ or if $\|\mathbf{r}\| > \sigma\sqrt{m+w}$.
  2. Let $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1] \in \mathbb{Z}_q^{n \times (m+w)}$. Return 1 if $\mathbf{B}_1 = \mathbf{A}_1 \cdot \mathbf{R}$ and $\mathbf{c} = \mathbf{A}_0 \cdot \boldsymbol{\mu} + \mathbf{B} \cdot \mathbf{r}$.
- $\mathbf{C} \leftarrow \mathbb{S}\mathsf{Commit}(mpk; (\mathbf{R}, \mathbf{r}))$: Given $\mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m \times w}, \sigma}$ and $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^{m+w}, \sigma}$, compute the matrix $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{G} - \mathbf{A}_1 \cdot \mathbf{R}] \in \mathbb{Z}_q^{n \times (m+w)}$ and $\mathbf{c} = \mathbf{B} \cdot \mathbf{r} \in \mathbb{Z}_q^n$. Output $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times w}$, where $\mathbf{B}_1 = \mathbf{G} - \mathbf{A}_1 \cdot \mathbf{R}$. Note that matrix $\mathbf{R}$ is a trapdoor for $\mathbf{B}$.
- $\tau \leftarrow \mathbb{S}\mathsf{Open}(mpk, \boldsymbol{\mu}, \mathsf{flag}; (\mathbf{R}, \mathbf{r}))$:
  - If $\mathsf{flag} = \mathbb{S}$, we must have $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) = (\mathbf{B} \cdot \mathbf{r}, \mathbf{G} - \mathbf{A}_1 \cdot \mathbf{R})$. Compute $\mathbf{c}' = \mathbf{c} - \mathbf{A}_0 \cdot \boldsymbol{\mu}$ and sample $\mathbf{r}' \leftarrow \mathsf{SampleD}(\mathbf{R}, \mathbf{B}, \mathbf{c}', \sigma)$ (Lemma 4).
    Then, output $\tau = \mathbf{r}' \in \mathbb{Z}^{m+w}$, which satisfies $\mathbf{c} = \mathbf{A}_0 \cdot \boldsymbol{\mu} + \mathbf{B} \cdot \mathbf{r}'$ and $\|\mathbf{r}'\| \leq \sigma\sqrt{m+w}$ with overwhelming probability (Lemma 1).
  - If $\mathsf{flag} = \mathbb{H}$, output $\tau = \mathbf{r} \in \mathbb{Z}^{m+w}$.
- $\mathbb{S}\mathsf{Verify}(mpk, \boldsymbol{\mu}, \mathbf{C}, \tau)$: Let $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times w}$ and $\tau = \mathbf{r} \in \mathbb{Z}^{m+w}$ and define $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1] \in \mathbb{Z}_q^{n \times (m+w)}$. Return 1 if $\mathbf{c} = \mathbf{A}_0 \cdot \boldsymbol{\mu} + \mathbf{B} \cdot \mathbf{r}$ and $\|\mathbf{r}\| \leq \sigma\sqrt{m+w}$. Otherwise, return 0.

- $\mathbf{C} \leftarrow \mathsf{MFake}(mpk; (\mathbf{R}, \mathbf{r}))$: Given $\mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m \times w}, \sigma}$ and $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^{m+w}, \sigma}$, compute $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1] \in \mathbb{Z}_q^{n \times (m+w)}$, where $\mathbf{B}_1 = \mathbf{A}_1 \cdot \mathbf{R}$, and compute $\mathbf{c} = \mathbf{B} \cdot \mathbf{r}$. Output $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1)$.
- $\pi \leftarrow \mathbb{H}\mathsf{Equivocate}(msk, \boldsymbol{\mu}; (\mathbf{R}, \mathbf{r}))$: Let $msk = \mathbf{T} \in \mathbb{Z}^{\bar{m} \times w}$ and let the fake commitment be $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) = (\mathbf{B} \cdot \mathbf{r}, \mathbf{A}_1 \cdot \mathbf{R})$, where $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{A}_1 \cdot \mathbf{R}]$. Compute $\mathbf{c}' = \mathbf{c} - \mathbf{A}_0 \cdot \boldsymbol{\mu}$. Then, extend $\mathbf{T}$ into a trapdoor $\mathbf{T}_{\mathbf{B}}$ for the matrix $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{A}_1 \cdot \mathbf{R}]$ and sample $\mathbf{r}' \leftarrow \mathsf{SampleD}(\mathbf{T}_{\mathbf{B}}, \mathbf{B}, \mathbf{c}', \sigma)$. Output $\pi = (\mathbf{R}, \mathbf{r}') \in \mathbb{Z}^{m \times w} \times \mathbb{Z}^{m+w}$.
- $\tau \leftarrow \mathbb{S}\mathsf{Equivocate}(msk, \boldsymbol{\mu}; (\mathbf{R}, \mathbf{r}))$: Let $msk = \mathbf{T} \in \mathbb{Z}^{\bar{m} \times w}$ and let the fake commitment be $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) = (\mathbf{B} \cdot \mathbf{r}, \mathbf{A}_1 \cdot \mathbf{R})$, where $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{A}_1 \cdot \mathbf{R}]$. Compute $\mathbf{c}' = \mathbf{c} - \mathbf{A}_0 \cdot \boldsymbol{\mu}$. Then, extend $\mathbf{T}$ into a trapdoor $\mathbf{T}_{\mathbf{B}}$ for the matrix $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{A}_1 \cdot \mathbf{R}]$ and sample $\mathbf{r}' \leftarrow \mathsf{SampleD}(\mathbf{T}_{\mathbf{B}}, \mathbf{B}, \mathbf{c}', \sigma)$. Output $\tau = \mathbf{r}' \in \mathbb{Z}^{m+w}$.
- $(\mathbf{R}', \mathbf{r}') \leftarrow \mathsf{FakeExplain}(msk; (\mathbf{R}, \mathbf{r}))$: Given $msk = \mathbf{T} \in \mathbb{Z}^{\bar{m} \times w}$ together with a Gaussian matrix $\mathbf{R} = [\mathbf{r}_1 \mid \ldots \mid \mathbf{r}_w] \hookleftarrow D_{\mathbb{Z}^{m \times w}, \sigma}$ and a vector $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^{m+w}, \sigma}$ such that $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) = (\mathbf{B} \cdot \mathbf{r}, \mathbf{A}_1 \cdot \mathbf{R})$ is a fake commitment, set $\mathbf{r}' = \mathbf{r}$ and use the trapdoor $\mathbf{T}$ for $\mathbf{A}_1$ to sample a small-norm $\mathbf{R}' = [\mathbf{r}'_1 \mid \ldots \mid \mathbf{r}'_w]$ such that $\mathbf{A}_1 \cdot \mathbf{R}' = \mathbf{G} - \mathbf{A}_1 \cdot \mathbf{R}$. To do this, let $\mathbf{G} = [\mathbf{g}_1 \mid \ldots \mid \mathbf{g}_w]$, and for each $i \in [w]$, sample $\mathbf{r}'_i \leftarrow \mathsf{SampleD}(\mathbf{T}, \mathbf{A}_1, \mathbf{g}_i - \mathbf{A}_1 \cdot \mathbf{r}_i)$.
  Then, output $(\mathbf{R}', \mathbf{r}')$ which satisfy $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) = (\mathbf{B} \cdot \mathbf{r}', \mathbf{G} - \mathbf{A}_1 \cdot \mathbf{R}')$.

5.2 Analysis

We prove that the trapdoor mercurial commitment scheme described in Section 5.1 satisfies the correctness and security properties defined in Section 2.1.

Correctness. By Lemma 1, with overwhelming probability, samples from discrete Gaussian distributions $D_{\mathbb{Z}^m, \sigma}$ and $D_{\mathbb{Z}^{m+w}, \sigma}$ have their Euclidean norms bounded by $\sigma\sqrt{m}$ and $\sigma\sqrt{m+w}$, respectively. Moreover, the outputs of $\mathsf{SampleD}$ are statistically close to discrete Gaussian samples, by Lemma 4. Therefore, if proofs π and τ are generated as in Section 5.1, then they should pass the verifications for Euclidean norms performed by algorithms $\mathbb{H}\mathsf{Verify}$ and $\mathbb{S}\mathsf{Verify}$. Note further that the equations modulo q verified by these algorithms must hold by construction. As a result, the scheme is correct with overwhelming probability.

Security. In the following lemmas, we show that the proposed scheme satisfies mercurial-binding under the SIS assumption, and HH, HS, SS and SE equivocation in the statistical sense.

Lemma 5. *The scheme is mercurial-binding under the* $\mathsf{SIS}_{n,m,q,\beta}$ *assumption, with* $\beta = \sigma \cdot (l\sqrt{\bar{m}} + \sqrt{\sigma m \bar{m}(\sigma^2 w^3 + 2m)})$.

Proof. Since the scheme is a proper mercurial commitment (i.e., hard openings contain their corresponding soft opening as a proper subset), we only need to consider the hard-soft case. Towards a contradiction, let us assume that the adversary can come up with a commitment $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times w}$ which it can hard-open to a message $\boldsymbol{\mu}$ and soft-opened to a different message $\boldsymbol{\mu}'$. This means that the adversary can output $(\boldsymbol{\mu}, \mathbf{R}, \mathbf{r}) \in \{0,1\}^l \times \mathbb{Z}^{m \times w} \times \mathbb{Z}^{m+w}$ and $(\boldsymbol{\mu}', \mathbf{r}') \in \{0,1\}^l \times \mathbb{Z}^{m+w}$ such that $\mathbf{B}_1 = \mathbf{A}_1 \cdot \mathbf{R}$ and

$$\mathbf{c} = \mathbf{A}_0 \cdot \boldsymbol{\mu} + [\mathbf{A}_1 \mid \mathbf{A}_1 \cdot \mathbf{R}] \cdot \mathbf{r} = \mathbf{A}_0 \cdot \boldsymbol{\mu}' + [\mathbf{A}_1 \mid \mathbf{A}_1 \cdot \mathbf{R}] \cdot \mathbf{r}'. \qquad (1)$$

Assuming that such a mercurial-binding adversary $\mathcal{A}$ exists, we can build a $\mathsf{SIS}_{n,m,q,\beta}$ solver $\mathcal{B}$ which takes as input a $\mathsf{SIS}_{n,m,q,\beta}$ instance $\mathbf{A} \in \mathbb{Z}_q^{n \times \bar{m}}$ and finds a non-zero vector $\mathbf{v}^\star \in \mathbb{Z}^{\bar{m}}$ of $\Lambda^\perp(\mathbf{A})$ such that $\|\mathbf{v}^\star\| \leq \beta$. To this end, $\mathcal{B}$ samples $\mathbf{R}_0 \hookleftarrow D_{\mathbb{Z},\sigma}^{\bar{m} \times l}$, $\mathbf{R}_1 \hookleftarrow D_{\mathbb{Z},\sigma}^{\bar{m} \times m}$ and defines

$$\mathbf{A}_0 = \mathbf{A} \cdot \mathbf{R}_0 \ \in \mathbb{Z}_q^{n \times l}, \qquad\qquad \mathbf{A}_1 = \mathbf{A} \cdot \mathbf{R}_1 \ \in \mathbb{Z}_q^{n \times m}.$$

Note that, by Lemma 2, matrices $\mathbf{A}_0$ and $\mathbf{A}_1$ are statistically close to the distributions $U(\mathbb{Z}_q^{n \times l})$ and $U(\mathbb{Z}_q^{n \times m})$, respectively. The adversary $\mathcal{A}$ is given $mpk = (\mathbf{A}_0, \mathbf{A}_1)$ and, assuming that it can output $(\boldsymbol{\mu}, \mathbf{R}, \mathbf{r})$ and $(\boldsymbol{\mu}', \mathbf{r}')$ satisfying (1) for distinct $\boldsymbol{\mu} \neq \boldsymbol{\mu}'$, we have

$$\mathbf{A}_0 \cdot (\boldsymbol{\mu} - \boldsymbol{\mu}')) = \mathbf{A}_1 \cdot [\mathbf{I}_m \mid \mathbf{R}] \cdot (\mathbf{r}' - \mathbf{r}) \mod q.$$

This implies that

$$\mathbf{v}^\star = \mathbf{R}_0 \cdot (\boldsymbol{\mu} - \boldsymbol{\mu}') + \mathbf{R}_1 \cdot [\mathbf{I}_m \mid \mathbf{R}] \cdot (\mathbf{r} - \mathbf{r}') \quad \in \mathbb{Z}^{\bar{m}} \qquad (2)$$

is a short vector of $\Lambda^\perp(\mathbf{A})$ with norm $\|\mathbf{v}^\star\| \leq \sigma \cdot (l\sqrt{\bar{m}} + \sqrt{\sigma m \bar{m}(\sigma^2 w^3 + 2m)})$. Moreover, we claim that it is non-zero with overwhelming probability. Indeed, $(\boldsymbol{\mu} - \boldsymbol{\mu}') \in \{-1,0,1\}^l$ has at least one non-zero coordinate by hypothesis. Given that the columns of $\mathbf{R}_0$ have at least $\Omega(n)$ bits of min-entropy conditionally on $\mathbf{A}_0 = \mathbf{A} \cdot \mathbf{R}_0$ (by Lemma 2 and Lemma 3), the product $\mathbf{R}_0 \cdot (\boldsymbol{\mu} - \boldsymbol{\mu}')$ is a linear combination (with coefficients in $\{-1,0,1\}$) of the columns of $\mathbf{R}_0$ which contains a completely unpredictable term. Hence, the right-hand-side member of (2) can only cancel over $\mathbb{Z}^{\bar{m}}$ with negligible probability. □

Lemma 6. *The scheme provides HH, HS, SS and SE equivocation in the statistical sense.*

Proof. For any message $\boldsymbol{\mu}$, we show that the distribution of fake commitments and their hard equivocations to $\boldsymbol{\mu}$ is statistically close to that of hard commitments and their hard openings to $\boldsymbol{\mu}$.

We note that $\mathbf{B}_1$ is generated in the same way in both fake and hard commitments. Moreover, since $\mathbf{A}_1$ is statistically uniform over $\mathbb{Z}_q^{n \times m}$, Lemma 2 implies that the distribution $\{(\mathbf{A}_1, \mathbf{B}_1) = (\mathbf{A}_1, \mathbf{A}_1 \cdot \mathbf{R}) \mid \mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m \times w},\sigma}\}$ is statistically close to the distribution $U(\mathbb{Z}_q^{n \times m}) \times U(\mathbb{Z}_q^{n \times w})$, meaning $\mathbf{B} \sim U(\mathbb{Z}_q^{n \times (m+w)})$ in both hard and fake commitments. By Lemma 2, we find that the distribution of fake commitments $(\mathbf{c}, \mathbf{B}_1)$, which is given by $\{([\mathbf{A}_1 \mid \mathbf{B}_1] \cdot \mathbf{r}, \mathbf{B}_1) \mid \mathbf{r} \hookleftarrow D_{\mathbb{Z}^{m+w},\sigma}\}$, is in turn statistically close to $U(\mathbb{Z}_q^n) \times U(\mathbb{Z}_q^{n \times (m+w)})$. This implies that the distribution of fake commitments remains statistically unchanged if we compute $\mathbf{c}$ as $\mathbf{c} = \mathbf{A} \cdot \boldsymbol{\mu} + \mathbf{B} \cdot \mathbf{r}$ instead of $\mathbf{c} = \mathbf{B} \cdot \mathbf{r}$. We call ideal_1 this modification of the ideal experiment. Moreover, by Lemma 2 again, we know that, for any statistically uniform matrix $\mathbf{A} \sim U\big(\mathbb{Z}_q^{n\times(m+w)}\big)$, the distribution

$$\Big\{(\mathbf{A}, \mathbf{A}\cdot\mathbf{r}, \mathbf{r}) \in \mathbb{Z}_q^{n\times(m+w)} \times \mathbb{Z}_q^n \times \mathbb{Z}^{m+w} \mid \mathbf{r} \hookleftarrow D_{\mathbb{Z}^{m+w},\sigma}\Big\} \qquad (3)$$

is statistically close to

$$\Big\{(\mathbf{A}, \mathbf{u}, \mathbf{r}) \in \mathbb{Z}_q^{n\times(m+w)} \times \mathbb{Z}_q^n \times \mathbb{Z}^{m+w} \mid \mathbf{u} \hookleftarrow U(\mathbb{Z}_q^n), \mathbf{r} \hookleftarrow D_{\Lambda^{\mathbf{u}}(\mathbf{A}),\sigma}\Big\}. \qquad (4)$$

Consequently, we can modify ideal_1 by changing the way to equivocate the fake commitment. Instead of using extending $\mathbf{T}$ into a trapdoor for $\mathbf{B} = [\mathbf{A}_1 \mid \mathbf{B}_1]$ and using it to sample $\mathbf{r}$ in a coset of the lattice $\Lambda^\perp(\mathbf{B})$, we just reveal the vector $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^{m+w},\sigma}$ that was used to compute $\mathbf{c} = \mathbf{A}\cdot\boldsymbol{\mu} + \mathbf{B}\cdot\mathbf{r}$. If we call this experiment ideal_2, we find it statistically indistinguishable from the ideal experiment thanks to the statistical closeness of (3)-(4). We observe that ideal_2 is nothing but the real HH equivocation experiment since $\mathbf{B}_1$ is generated in the same way in both experiments. This shows the HH equivocation property. The HS and SS equivocation properties can be shown in a completely similar way.

As for the SE equivocation property, it follows from two observations. First, Lemma 2 implies that the distributions

$$D_{\mathsf{fake}} := \big\{\mathbf{A}_1\cdot\mathbf{R} \mid \mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m\times w},\sigma}\big\}, \quad D_{\mathsf{soft}} := \big\{\mathbf{G} - \mathbf{A}_1\cdot\mathbf{R}' \mid \mathbf{R}' \hookleftarrow D_{\mathbb{Z}^{m\times w},\sigma}\big\}$$

are both statistically close to $U(\mathbb{Z}_q^{n\times w})$. Hence, the adversary's view remains statistically the same if we generate fake commitments by sampling $\mathbf{B}_1$ from D_{soft} instead of D_{fake} in the ideal experiment. Moreover, since distributions (3) and (4) are statistically close, $\mathcal{A}$'s view remains statistically the same after modification. instead of using the trapdoor $\mathbf{T}$ of $\Lambda^\perp(\mathbf{A}_1)$, we reveal the Gaussian matrix $\mathbf{R}'$, used to get $\mathbf{B}_1 = \mathbf{G} - \mathbf{A}_1\cdot\mathbf{R}'$ after sampling $\mathbf{R}' \hookleftarrow D_{\mathbb{Z}^{m\times w},\sigma}$. With this, the result is identical to the real game, proving the SE property. □

5.3 Remarks

The scheme from Section 5.1 produces commitments of the form $\mathbf{C} = (\mathbf{c}, \mathbf{B}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{n\times w}$, and thus, have length $k = n(w+1)\lceil \log q \rceil$ bits. Its message space is $\mathcal{M} = \{0,1\}^l$, where l can vary depending on the context.

The scheme leads to a lattice-based ZK-EEDB system, following the constructions of Sections 3 and 4. In this system, the following 4 different message lengths, $\{l_1, l_2, l_3, l_4\}$, are considered.

1. At leaves of the first tree, we commit to values of bit-length $l_1 = \ell$.
2. At non-leaf nodes in both trees, since we commit to 2 commitment strings, we work with message length $l_2 = 2k$.
3. At leaves of the second tree, we store commitments to $\mathsf{D}^{-1}(y)$, which is a commitment string of bit-length $l_3 = k$.
4. When building a commitment of $\mathsf{D}_y^{-1} = \{(x,1) \mid (x,y) \in \mathsf{D}\}$, we also work with message length $l_4 = 1$.

To handle these message lengths, we need only adjust the number of columns in $\mathbf{A}_0 \in \mathbb{Z}_q^{n\times l}$, with $l = \max\{l_1, l_2, l_3, l_4\}$. For each $i \in [4]$, we use $\mathbf{A}_{0,i} \in \mathbb{Z}_q^{n\times l_i}$, the matrix that is the first l_i columns of $\mathbf{A}_0$, to commit to a length-l_i message.

A description of an authentication path with its commitment strings requires $\zeta = \mathcal{O}(l \cdot k)$ bits, which is $\widetilde{\mathcal{O}}(\lambda^3)$ when $l = \mathcal{O}(\lambda)$. Fortunately, this can be greatly reduced if the TMC scheme is adapted to the ring setting. As shown by Micciancio and Peikert [22] and later by Ducas and Micciancio [8], with appropriate choice of parameters, all the lattice-based cryptographic ingredients of Section 2.3 can be adapted to the ring setting. This lets us use $w = \mathcal{O}(\log q)$ (instead of $w = \mathcal{O}(n \log q)$), thereby reducing the commitment size and ζ by a factor of $\mathcal{O}(\lambda)$. We refer to the full version of this work for the details.

Acknowledgements. Part of this research was funded by Singapore Ministry of Education under Research Grant MOE2016-T2-2-014(S). Another part was funded by BPI-France in the context of the national project RISQ (P141580). This work was also supported in part by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701). Khoa Nguyen was also supported by the Gopalakrishnan – NTU Presidential Postdoctoral Fellowship 2018. Huaxiong Wang was also supported by the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability Research Centres Funding Initiative.

References

1. M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *STOC*, pages 103–112. ACM, 1988.
2. D. Catalano, Y. Dodis, and I. Visconti. Mercurial commitments: Minimal assumptions and efficient constructions. In *TCC 2006*, 2006.
3. D. Catalano and D. Fiore. Vector commitments and their applications. In *PKC*, 2013.
4. D. Catalano, D. Fiore, and M. Messina. Zero-knowledge sets with short proofs. In *Eurocrypt*, 2008.
5. M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. In *EUROCRYPT 2005*, 2005.
6. M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. *J. of Cryptology*, 2013.
7. M. Chase and I. Visconti. Secure database commitments and universal arguments of quasi knowledge. In *Crypto*, 2012.
8. L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014*, 2014.
9. R. Gennaro and S. Micali. Independent zero-knowledge sets. In *ICALP*, 2006.
10. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *STOC 2008*, 2008.
11. E. Ghosh, O. Ohrimenko, and R. Tamassia. Verifiable order queries and order statistics on a list in zero-knowledge. In *ACNS*, 2015.
12. E. Ghosh, O. Ohrimenko, and R. Tamassia. Efficient verifiable range and closest point queries in zero-knowledge. *PoPETs*, 2016(4), 2016.

13. V. Goyal, R. Ostrovsky, A. Scafuro, and I. Visconti. Black-box non-black-box zero knowledge. In *STOC*, 2014.
14. Y. Ishai, E. Kushilevitz, R. Ostrovksy, and A. Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, 2007.
15. A. Kate, G. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *Asiacrypt*, volume 6477 of *LNCS*, pages 177–194. Springer, 2010.
16. A. Kawachi, K. Tanaka, and K. Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT 2008*, 2008.
17. B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *TCC 2010*, 2010.
18. M. Liskov. Updatable zero-knowledge databases. In *Asiacrypt*, 2005.
19. V. Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In *Asiacrypt*, 2009.
20. R. C. Merkle. A Certified Digital Signature. In *CRYPTO*, volume 435 of *LNCS*, pages 218–238. Springer, 1989.
21. S. Micali, M. O. Rabin, and J. Kilian. Zero-knowledge sets. In *44th FOCS*, 2003.
22. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, 2012.
23. D. Micciancio and C. Peikert. Hardness of SIS and LWE with Small Parameters. In *CRYPTO 2013*, 2013.
24. D. Micciancio and O. Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM Journal on Computing*, 37(1), 2007.
25. D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO 2001*, 2001.
26. M. Naor and K. Nissim. Certificate revocation and certificate update. In *7th USENIX Security Symposium*, 1998.
27. M. Naor and A. Ziv. Primary-secondary-resolver membership proof systems. In *TCC*, 2015.
28. R. Ostrovsky, C. Rackoff, and A. Smith. Efficient consistency proofs for generalized queries on a committed database. In *ICALP 2004*, 2004.
29. D. Papadopoulos, S. Papadopoulos, and N. Triandopoulos. Taking authenticated range queries to arbitrary dimensions. In *ACM-CCS*, 2014.
30. C. Papamanthou, R. Tamassia, and N. Triandopoulos. Optimal verification of operations on dynamic sets. In *Crypto*, 2011.
31. M. Prabhakaran and R. Xue. Statistically hiding sets. In *CT-RSA*, 2009.
32. R. Tamassia. Authenticated data structures. In *European Symp. on Algorithms (ESA)*, 2003.