Secure Database Commitments and Universal Arguments of Quasi Knowledge

Melissa Chase and Ivan Visconti

[1] Microsoft Research, Redmond, USA
melissac@microsoft.com
[2] Dipartimento di Informatica, University of Salerno, ITALY
visconti@dia.unisa.it

Abstract. In this work we focus on a simple database commitment functionality where besides the standard security properties, one would like to hide the size of the input of the sender. Hiding the size of the input of a player is a critical requirement in some applications, and relatively few works have considered it. Notable exceptions are the work on zero-knowledge sets introduced in [14], and recent work on size-hiding private set intersection [1]. However, neither of these achieves a secure computation (i.e., a reduction of a real-world attack of a malicious adversary into an ideal-world attack) of the proposed functionality.

The first result of this submission consists in defining "secure" database commitment and in observing that previous constructions do not satisfy this definition. This leaves open the question of whether there is any way this functionality can be achieved.

We then provide an affirmative answer to this question by using new techniques that combined together achieve "secure" database commitment. Our construction is in particular optimized to require only a constant number of rounds, to provide non-interactive proofs on the content of the database, and to rely on the existence of a family of CRHFs. This is the first result where input-size hiding secure computation is achieved for an interesting functionality and moreover we obtain this result with standard security (i.e., simulation in expected polynomial time against fully malicious adversaries, without random oracles, without non-black-box extraction assumptions, without hardness assumptions against super-polynomial time adversaries).

A key building block in our construction is a universal argument enjoying an improved proof of knowledge property, that we call quasi-knowledge. This property is significantly closer to the standard proof of knowledge property than the weak proof of knowledge property satisfied by previous constructions.

Keywords: ZK sets, universal arguments, input-size hiding security.

1 Introduction

Secure computation. The standard notion of "security" for any multi-party computation [10] involves describing an ideal model where parties have access to a trusted functionality which will carry out the desired computation without revealing any information about the parties' secret inputs and outputs. Then, very informally, given a communication network in the real world, without any trusted functionality, a protocol π securely realizes a multi-party computation if any successful attack by an adversary of π can be translated into a successful attack in the ideal world. Since the latter is trivially secure this means π is secure in the real world as well.

Therefore, the real-world/ideal-world paradigm [10] allows one to prove the security of protocols in a simulation-based fashion, with strong security guarantees. This turns out to be very useful when protocols are used as subprotocols and provides robust security when the ideal functionality is correctly designed. All the fundamental primitives have been defined in terms of ideal functionalities and thus they can be securely realized in the real-world/ideal-world paradigm using the general results of [10].

However the current state-of-the art still does not properly address the issue of considering *the size of the input* of a player as information to be kept private. Here we consider the problem of hiding the input size in one of the most basic primitives: a commitment scheme.

Secure Database Commitments. Here we address the case where a player wants to commit to a large set of data, and then to partially open that commitment, both without revealing the size of the committed set. More specifically, we consider the setting where a party (the sender) may want to commit to a large elementary database composed by key-value pairs, without revealing its size. Then in the opening phase, the sender might want to reveal part of his database one item at a time depending on queries of the receiver. In particular, the receiver will ask for some keys, and the sender wants to convince the receiver of the value associated to each requested key. These partial opening queries should reveal no information about the rest of the database, including the size.

The main question: feasibility of Secure Database Commitments. Following the above discussion, and the fact that we have general results for achieving secure multi-party computation, One could ask the following natural question: "Why should we focus on the design of protocols for secure database commitments if we can use the known constructions for secure two-party computation?". There is a crucial difference between this problem, and the one considered in [10] and in all subsequent work (to the best of our knowledge). Our notion of secure database commitment critically requires that the size of the prover's input must be hidden. In contrast, the [10] definition, according to [9] and to the known constructions, allows a party either at the beginning of or during the computation, to learn the size of the other party's input. This information is actually revealed in all known protocols, mainly because many of the tools used to allow extraction of the adversary's implicit input (e.g. zero-knowledge proofs of knowledge) have communication that depends directly on the size of the input.

Consequently, traditional results for secure two-party computation cannot be used to obtain secure database commitments, and we need to develop new tools and a significantly new approach. This presents the following interesting open problem: *Is the notion of secure database commitment and more generally a meaningful notion of input-size hiding secure computation achievable?*

Difficulties in achieving input-size hiding secure computation with standard assumptions. We stress that the main challenge in size-hiding secure computation is as follows: when proving security against a real world adversarial prover, we need to design a simulator which can extract the input from the adversary so that it can provide it to the ideal functionality. At the same time, we can not assume any fixed polynomial upper bound for the size of the set (the protocol should work for *any* polynomial sized input), and we can not allow the amount of communication to vary with the size of the input.

The real difficulties lie in obtaining a standard security notion (showing an efficient ideal adversary from an efficient real-world adversary), therefore avoiding controversial (sometimes not even falsifiable) conjectures such as random oracles, non-black-box extraction assumptions (e.g., the Diffie-Hellman knowledge of exponent assumption and its variations [11]), complexity leveraging (i.e., hardness assumptions against super-polynomial time adversaries) and so on.

Relationship to Zero-Knowledge Sets. We note that the requirements described for secure database commitments are essentially those given in [14] where Micali, Rabin and Kilian introduced the concept of a zero-knowledge set (ZKS). This primitive allows a party to first commit to a database, and later to answer queries on that database and to prove that the responses are consistent with the original commitment, as in the secure database commitments described above.

However, the definition given by [14] and by all subsequent papers is a *property-based definition* that requires: 1) soundness: the commitment should be binding, i.e., for each query there should be only one response for which the prover can give a convincing proof; and 2) zero-knowledge: both the commitment and the proofs should not reveal any information about the rest of the database (beyond the information that is asked in the query), not even the number of elements it contains.

Furthermore, we argue that the constructions they provide (and later constructions for ZKS, see [7,5,8,6,15,13]) do not satisfy a typical real-world/ideal-world definition in the spirit of "secure computation". To see this, note that all previous schemes for ZKS included a non-interactive commitment of the set. As mentioned above, we do not consider non-black-box extraction assumptions; moreover, standard non-black-box techniques introduced in [2] so far have been successfully used only in conjunction with interaction.

However, we argue that black box extraction is impossible for a scheme which is size hiding and has a non-interactive commitment phase. This follows because such a scheme must allow for sets of *any* polynomial size, which means there will be at least $2^{\mathsf{superpoly}(k)}$ possible sets; at the same time, the length of the non-interactive commitment must be bounded by a fixed polynomial in k. Thus, a simple counting argument shows that there is no way (based on standard assumptions and security notions) that a simulator can correctly identify the committed set given only the single commitment message. Note that this argument also holds in the CRS model, as long as the CRS is of length polynomial in k, as even in this case the simulator still gets only $\mathsf{poly}(k)$ bits of information to identify one database out of $2^{\mathsf{superpoly}(k)}$. Therefore, no previous construction of zero-knowledge sets can be proved to be a secure realization of the database commitment functionality (with a black box simulator).

Looking ahead, in our construction we will avoid these limitations by allowing an interactive commitment phase. The possibility of using interaction was already mentioned in some previous work, but only for the query/proof phase, thus without making any contribution towards addressing this extraction problem. Instead, our goal is to use interaction in the commitment phase, still keeping the query/answer phase essentially non-interactive.

Other related work. Recently similar issues have been considered by Ateniese et al. in [1] for the problem of Private Set Intersection. However their solution uses random oracles, and obtains security with respect to semi-honest adversaries only. The goal of our work is to obtain security against malicious players as required in secure computation, and we will obtain this result in the standard model using only standard complexity-theoretic assumptions.

Timing attacks. We note that there may be some cases in which any attempt to hide the size of inputs is vulnerable to timing attacks as discussed in [12] (in which an adversary can guess the size of the input depending on the amount of the time required to perform computations). However, there are many settings where these attacks will have only limited impact. Notice that since the adversary does not necessarily know the amount of resources of the other player (the committer could perform his computation by distributing the workload among computing devices in number proportional to the size of the input) the timing may in fact give very little information.

1.1 Our Results

In this work we first put forth the notion of secure database commitment. Following the traditional notion of "secure computation" we define the natural ideal functionality for database commitment $\mathcal{F}_{DbCom}$ and observe that it implies all required security guarantees needed by zero-knowledge sets.

We then present a constant-round protocol that securely realizes the $\mathcal{F}_{DbCom}$ functionality. The protocol is interactive, and is secure in the standard model (i.e., we do not require any set-up assumptions) based on the existence of families of collision-resistant hash functions (CRHFs, for short) (notice that CRHFs are implied by the existence of ZK sets [5]).

We stress that our protocol has optimal amortized round complexity, as queries and answers are essentially non-interactive. In addition, the use of the real-world/ideal-world paradigm, and the simulation in polynomial time should make it much easier to use this primitive as part of a larger protocol.

Our construction is based on a special universal argument that enjoys a new proof of knowledge property which is closer to the traditional proof of knowledge property. We define this new property, give a construction which satisfies it based on CRHFs, and show how it can be used to implement secure database commitments. (However, we stress that there were many other subtle issues to address, and our stronger notion of universal argument alone is not sufficient for directly obtaining input-size hiding secure computation.)

Techniques. As described above, the biggest challenge is in defining a simulator that can extract a witness whose length can be any polynomial, from only a fixed polynomial amount of communication. Note that the standard approach does not work in this setting: it might seem that a simple solution would be to ask the sender to give a commitment to the database, and an interactive zero-knowledge proof of knowledge of the database contained in the provided commitment. However, in general such proofs may reveal the size of the witness, which in this case means the size of the database.

Instead, we use a different tool, namely a witness indistinguishable universal argument of quasi knowledge (UAQK). A universal argument is a (computationally-sound) proof system which guarantees that the communication is only proportional to the size of the statement. At the same time, it guarantees that the honest prover can run in time polynomial in the size of the witness, which is the best that we can hope for.

Universal arguments were introduced by Barak [2] as part of the design of a non-black-box simulator. Barak showed a construction based on CRHFs which satisfied a weak proof of knowledge property. However, there are some inherent challenges in defining a standard proof of knowledge for universal arguments, and the extraction guarantees of his definition do not seem to be sufficient for our application. To deal with this we define a new proof of knowledge property which we call "quasi knowledge" which provides a functionality somewhat closer to that of a standard proof of knowledge. We show that it can be used in our application to implement the traditional commitment and proof of knowledge strategy described above. (Of course we are glossing over many issues here, and the application is not straightforward at all - see Section 4 for details.)

Finally, we note that this is of additional interest, because we use universal arguments for a completely different purpose than the one that motivated Barak's original construction [2], which was the design of a non-black-box simulator.

Universal Arguments: from weak proof of knowledge to proof of quasi knowledge. The standard proof of knowledge property guarantees that if a prover can convince a verifier of the truthfulness of a theorem with some probability, then one can efficiently obtain an NP witness for that theorem with the same probability (up to negligible factors).

When one focuses instead on universal languages (i.e., when one would like to prove that a Turing machine accepts the provided input within a given — not necessarily polynomial — number of steps), then one can not always efficiently obtain a corresponding witness since its length can be superpolynomial. In this case a restricted proof of knowledge property might instead focus on extracting an implicit representation of the witness, in the form of a polynomial-sized circuit that when evaluated on input i provides the i-th bit of the witness.

Unfortunately there is no known construction of a universal argument with such a property. The current state of the art, [4], shows how to get a *weak* proof of knowledge property that includes one more restriction: when a prover proves a theorem with probability $1/q$, one can efficiently get an implicit representation of a circuit with probability $1/q'$ that is polynomially related to $1/q$. This essentially means that one can efficiently get a *candidate* implicit representation of a witness, with non-negligible probability when $1/q$ is non-negligible. However, there is no guarantee that the candidate implicit representation is actually correct (one can not simply test the circuit asking for all bits of the witness since the size of a witness can be superpolynomial).

Furthermore, there is an additional subtlety in the way this weak proof of knowledge property is defined. For any polynomial q, there is guaranteed to be a polynomial time extractor that can extract candidate representations of the witness from any prover that proves a theorem with probability $1/q$, and this extractor is guaranteed to succeed with the related probability $1/q'$; however, the choice of this extractor and its running time may depend on q. This has two disadvantages: (1) in order to use an extractor, we must first determine which q we are interested in, and (2) an efficient extractor is required to exist only for polynomials q, while nothing is required when q is super polynomial. (In fact, in the construction given in [4], the running time of the extractor is roughly q^d where d is some constant greater than 1, therefore when q is superpolynomial the running time of the extractor increases quickly and its expected running time is not polynomial). As a result of these disadvantages, converting a weak proof of knowledge system into a proof system with the standard proof of knowledge property is very non-trivial, even when one is happy with the extraction of an implicit representation of the witness. (This is because the standard proof of knowledge property requires that, regardless of the success probability of the adversarial prover, the extractor must run in expected polynomial time.)

In this paper we remove the above additional restriction of the *weak* proof of knowledge property and replace it with a more standard proof of knowledge property, requiring that an extractor outputs in expected polynomial time a correct implicit representation of the witness. The only caveat is that, while we require that the extracted implicit representation must correspond to some true witness, we do allow it to contain a few errors, as long as those errors are unlikely to be noticed by any polynomial time process. (Note that if the witness is polynomial sized, then this still implies the standard proof of knowledge property.) We will say that an argument system is an argument of "quasi" knowledge if it enjoys this property.

Finally, we construct a constant-round witness-indistinguishable universal argument of quasi-knowledge under standard complexity-theoretic assumptions.

2 Universal Arguments of Quasi Knowledge

A universal argument is an interactive argument system for proving membership in **NEXP**. For the formal definition, see [4]. We will use the following result.

Theorem 1. ([4,3], informal) *Suppose there exists a hash function ensemble that is collision-resistant against polynomial-sized circuits. Then there exists a universal argument system that is constant-round, public-coin and witness indistinguishable.*

Moreover, there exists a construction which satisfies two additional properties. First, the construction of this extractor is parameterized by a lower bound $\alpha(n)$ on the success probability of the adversarial prover $P^\star_n$. The resulting extractor, that we denote by $E(\alpha(n), \cdot, \cdot)$ has the property that there is a fixed constant d such that, $E(\alpha(n), y, \cdot)$ has expected running time at most $(\frac{n}{\alpha(n)})^d$ for any n-bit instance y.

Furthermore, it has the property that there is a fixed known polynomial $q^\star$ such that for any polynomial-sized prover $P^\star_n$ that succeeds in causing verifier $V(y)$ to accept with probability $q(n)$ for n-bit instance y, and for any $\alpha < q(n)$, the probability that the extractor $E^{P^\star_n}(\alpha, y, \cdot)$ succeeds in extracting bits of a valid witness is at least $\frac{\alpha}{q^\star(n)}$. For a more formal statement, see the full version. For details of the construction and proof, see the proof of Lemma A.2.5 in [3].

2.1 A New Notion: Quasi-Knowledge

As described in Section 1.1, we aim to construct a universal argument with a more standard proof of knowledge property. The resulting proof of knowledge will resemble the standard proof of knowledge property with two exceptions.

The first is that the extractor will produce only an implicit representation of a witness. This is necessary since UAs are used to prove statements that are not necessarily in NP, therefore the length of the witness may not be polynomial, but still we want our extractor to run in (expected) polynomial time. We formalize this saying that the extractor will be a circuit which on input i will produce a candidate for the i-th bit of the witness w.

The second difference is that even when the extractor is successful, we do not require that the extracted witness be perfectly correct. We may allow some small fraction of the extracted bits to be incorrect, as long as this will be negligible when the proof is used in any polynomial-time process. We formalize this by saying that for any (potentially adversarially generated) polynomial sized set of indices I, it must be the case that with all but negligible probability, all of the associated bits produced by the extractor will be correct with respect to some valid witness $\hat{w}$.

Thus, our definition requires an extractor which satisfies two properties. The first is that, for any PPT (and potentially adversarial) sampling algorithm S which chooses a set of indices I, with all but negligible probability over the choice of random tape r, the extractor E_r with oracle access to $P^\star$ will output a set of bits $\{w_i\}_{i\in I}$ that are consistent with some valid witness $\hat{w}$. Note that we allow S to choose these indices adaptively, after querying E on indices of it's choice. This allows for the fact that in a larger application, which positions of the witness need to be extracted may depend on the values of the bits which have been seen so far.

The second property requires that the expected running time of E be within a polynomial factor of $\frac{1}{p(|y|)}$ where $p(|y|)$ is the success probability of the adversarial prover P^*. There is a slight issue here in that, even if the *expected* running time of $E(y,i)$ was guaranteed to be polynomial for each i, it might still be possible that for any choice of random tape r, an adversarial and adaptive sampling algorithm S could find at least one index that causes $E_r(y,i)$ to run in superpolynomial time. Thus we also require that the running time be independent of i; this implies that the expected running time of E on any set of inputs (even those that are chosen adversarially and adaptively) will also be polynomially related to $\frac{1}{p(|y|)}$. For our application this will be particularly useful, as it implies that E can be converted into a circuit which implicitly describes the witness and whose expected size is polynomially related to the positive success probability of P^* (and therefore the expected size of E is polynomial whenever P^* has non-negligible success probability). (We will see more details in Section 4.)

Definition of universal argument of quasi knowledge (UAQK). We construct a universal argument $\langle P(\cdot,\cdot), V(\cdot)\rangle$ such that the efficient verification, completeness and computational soundness properties hold as usual, but the weak proof of knowledge property is replaced as follows.

Definition 1. *A universal argument system $\langle P(\cdot,\cdot), V(\cdot)\rangle$ for the universal language L_U is a universal argument of* quasi knowledge *(UAQK) if there exists an algorithm E and a constant c such that for any polynomial-size circuit family $\{P^\star_n\}_{n\in N}$, there exists a negligible function ν such that for any $y \in L_U \cap \{0,1\}^n$, the following two properties hold.*

1. *If* $\mathrm{Prob}[\ \mathtt{out}_V(\langle P^\star_n, V(y)\rangle) = 1\ ]$ *is non-negligible, then for any polynomial-time sampling algorithm* S,

$$\mathrm{Prob}_r[\ I \leftarrow \mathsf{S}^{E_r^{P^\star_n}(y,\cdot)}(y); \{w_i \leftarrow E_r^{P^\star_n}(y,i)\}_{i\in I} :$$
$$\exists \hat{w} = \hat{w}_1,\dots \hat{w}_s, (\hat{w},y) \in R_U \wedge (w_i = \hat{w}_i\ \ \forall i \in I)\ ] \geq 1 - \nu(|y|).$$

   where the probability is over the choice of the coins r used by E. (Note that the same coins are used for all $i \in I$.)[3]

2. *The running time of $E^{P^\star_n}(y,i)$ is independent of the choice of i, and if we let* $p(|y|) = \mathrm{Prob}[\ \mathtt{out}_V(\langle P^\star_n, V(y)\rangle) = 1\ ] > 0$*, then the expected running time of $E^{P^\star_n}(y,\cdot)$ is $O(\frac{|y|^c}{p(|y|)})$, where again the expectation is over the choice of the coins r used by E.*

Note that if L_U is a language with polynomial-size witnesses, then this property implies the standard proof of knowledge property. The standard proof of knowledge extractor will first run $\langle P^\star_n, V(y)\rangle$ once: if V rejects, it will output $\perp$, otherwise it will choose a random string r to be used as randomness to run $E_r^{P^\star}(y,i)$ for each bit i of the witness, and then output the result. This extractor

[3] Here we assume for simplicity that if a witness w is expanded by appending any sequence of zeros, the resulting value is still a valid witness, so that $\hat{w}_i$ is always defined for all $i \in I$.

will have success probability negligibly far from $p|y|$, and expected running time $O(p(|y|) \cdot \frac{|y|^c}{p(|y|)}) = O(|y|^c)$, which is clearly polynomial.

Note also that we could have given a definition that explicitly requires the extractor to run in expected polynomial time, essentially resembling the definitions of the standard proof of knowledge and weak proof of knowledge properties. However, we prefer the above formulation as it makes clear that we can differentiate between (item 1) runs on which the extractor inadvertently produces a bad witness (recall that when the witness is not polynomial-sized, it may not be possible to efficiently identify invalid witnesses) and (item 2) runs on which the extractor aborts (e.g., when the interaction with $P_n^\star$ produces an invalid proof).

2.2 CRHFs ⇒ Constant-Round UAQK

Our approach will be to construct a UAQK out of a UA with the weak proof of knowledge property. As mentioned in Section 1.1, there are two difficulties when using the weak proof of knowledge property: (1) we must somehow estimate a lower bound on the success probability of the prover in order to determine which extractor to use, and (2) the running time of the extractor may grow faster than what we would like as compared with the success probability of the prover (here we will use the UA construction of [4], which gives an extractor with running time $O((\frac{|y|}{\alpha(|y|)})^d)$ where $\alpha(|y|)$ is a lower bound on the success probability of $P^\star$).

The first issue we will deal with within the design of our extractor; it essentially requires balancing the accuracy of the estimate with the need to compute it in expected polynomial time. In order to address the second issue, we will attempt to increase the success probability of the adversarial prover. Essentially, we will run many instances of that UA sequentially, and accept only if all instances are accepted by the verifier. Then, if the prover succeeds in all instances with probability p, we will argue that there must be at least one instance in which it succeeds with significantly higher probability. If we use this instance, then we can run the extractor with a higher lower bound, and thus obtain a more efficient extractor.

The resulting construction goes as follows. We first ask the prover to commit to a Merkle hash of its witness, and then we run the UA several times sequentially. In each UA, the prover proves both that the statement is true, and that the witness used is consistent with the given commitment, i.e. it proves weak knowledge of a witness and an authentication chain for each bit of the witness. (This will allow us to verify that parts of the witness are consistent with the initial commitment without having to extract the entire witness.)

The actual UAQK. Our construction uses as a subprotocol a universal argument $\langle P_{UA}(\cdot,\cdot), V_{UA}(\cdot)\rangle$ satisfying the conditions described in Theorem 1 for the universal language L_U. Let $\ell = d + 2$ where d is the value of the constant defined in Theorem 1 for this UA. We construct a UAQK $\langle P_{new}(y, w), V_{new}(y)\rangle$ for language L_U as depicted in Fig. 1.

Players: prover P_{new}, verifier V_{new}.
Common input: the statement $y \in L_U$, such that $|y| = n$.
Input of P_{new}: witness w for $y \in L_U$.
Tools: a family $\{\mathcal{H}_n\}$ of CRHFs, a statistically hiding trapdoor commitment scheme $(\mathsf{SHTGen}, \mathsf{SHCom}, \mathsf{SHVer}, \mathsf{SHTCom}, \mathsf{SHTDec})$, a constant-round statistical ZKAoK $\langle P_{szk}(\cdot,\cdot), V_{szk}(\cdot)\rangle$ for $\mathtt{NP}$, a constant-round WI universal argument $\langle P_{UA}(\cdot,\cdot), V_{UA}(\cdot)\rangle$ (as defined in Theorem 1) with parameter d.

1) $V_{new} \rightarrow P_{new}$: V_{new} picks $h \overset{R}{\leftarrow} \{\mathcal{H}_n\}$, computes $(\eta, t) \overset{R}{\leftarrow} \mathsf{SHTGen}(1^n)$ and sends (h, η) to P_{new}.
2) $V_{new} \leftrightarrow P_{new}$: V_{new} runs P_{szk} to prove to P_{new} running V_{szk} knowledge of a trapdoor t for η.
3) $P_{new} \rightarrow V_{new}$: P_{new} encodes the witness w as follows: Let $s = |w|$ be the length of the original witness w, and let $w = w_1, \dots, w_s$ be the bitwise representation of w. Then P_{new} uses h to form a Merkle hash tree over all the bits of w treating each bit as a leaf node in the tree. Let *root* be the root of this tree, and for bit w_i of witness w, let $auth_i$ be the appropriate authentication chain (i.e., all of the values on the path from w_i to *root*, and all of the siblings of those values).
   Next P_{new} computes $(C_s, \mathsf{dec}_s) \leftarrow \mathsf{SHCom}_\eta(s)$ and $(C_{root}, \mathsf{dec}_{root}) \leftarrow \mathsf{SHCom}_\eta(root)$.
   Finally, P_{new} forms the new witness $w' = s \circ \mathsf{dec}_s \circ root \circ \mathsf{dec}_{root} \circ w_1 \circ auth_1 \circ \dots \circ w_s \circ auth_s$. Now, let $y' = C_s \circ C_{root} \circ y$ and let L'_U be the language which accepts (y', w') iff $w' = s \circ \mathsf{dec}_s \circ root \circ \mathsf{dec}_{root} \circ w_1 \circ auth_1 \circ \dots \circ w_s \circ auth_s$ such that (1) $s < 2^{|y|}$, (2) (s, dec_s) is a valid decommitment for C_s and $(root, \mathsf{dec}_{root})$ is a valid decommitment for C_{root} with parameter η, (3) for all $i = 1 \dots s$, $auth_i$ is a valid authentication chain for w_i with respect to *root* and hash function h, and (4) $w = w_1 \dots, w_s$ is such that $(y, w) \in L_U$.
   P_{new} sends (C_s, C_{root}) to the verifier.
4) $P_{new} \leftrightarrow V_{new}$:
   For $j = 1, \dots, \ell = d + 2$, sequentially:
   4.j) P_{new} and V_{new} run the universal argument $\langle P_{UA}(\cdot,\cdot), V_{UA}(\cdot)\rangle$ for this new language L'_U. P_{new} will run $P_{UA}(w', y')$ while V_{new} will run $V_{UA}(y')$, where $y' = C_s \circ C_{root} \circ y$.
   V_{new} outputs 1 iff all instances of $V_{UA}(y')$ accept.

Fig. 1. Universal Argument of Quasi Knowledge $\langle P_{new}(\cdot,\cdot), V_{new}(\cdot)\rangle$.

2.3 Proving the Quasi-Knowledge Property

Intuition. At a high level, the proof proceeds as follows: Let p be the probability that the adversarial prover $P^\star$ convinces V_{new} to accept. Then for $j = 1, \dots, \ell$, let p_j be the probability that V_{new} accepts the j-th internal UA instance conditioned on the event that it accepted all previous instances. Then the key observation is that $p = \prod_{j=1}^{\ell} p_j$, so we are guaranteed that for some j^*, $p_{j^*} \geq p^{1/\ell}$. Now, if we could estimate p_{j^*}, and identify a UA instance where $P^\star$ succeeds with probability roughly p_{j^*}, then we could run the UA extractor with this lower bound, and obtain an extractor with success probability roughly $\frac{p^{1/\ell}}{q(|y|)}$ (here q is the polynomial referred to as $q^\star$ in the discussion in Theorem 1), and running time roughly $O((\frac{|y|}{p_{j^*}})^d) = O(\frac{|y|^d}{p^{d/\ell}})$. However, we need an extractor with overwhelming success probability, so we will run this extractor approximately $\frac{q(|y|)}{p^{1/\ell}}$ times, to ensure that at least one run will produce bits of a valid implicit witness.[4] The final result will have success probability nearly 1, and running time roughly $O(\frac{|y|^d}{p^{d/\ell}} \cdot \frac{q(|y|)}{p^{1/\ell}}) = O(\frac{|y|^d q(|y|)}{p^{(d+1)/\ell}}) = O(\frac{|y|^d q(|y|)}{p})$ (the last equality follows from our choice of ℓ). [5]

The main challenge in this process is finding a UA instance where $P^\star$ has success probability roughly $p^{1/\ell}$, and estimating this probability. Essentially, for each j, we want to find a starting state where $P^\star$ is about to begin the j-th UA, and where $P^\star$'s success probability in that proof is roughly p_j; then we need to identify and take the best of these states (i.e., p_{j^*}). We do this as follows: First, for each j, we record many states for $P^\star$ where $P^\star$ has successfully completed the first $j-1$ UAs and has a reasonable chance of completing the next UA. In the full version we show that with overwhelming probability, one of these states will be such that $P^\star$ has probability at least $p_j/2$ of successfully completing the next UA, and at the same time that the time required to collect all these states is at most $O(\mathsf{poly}(|y|)/p)$. Next, we attempt to identify one such state, and to estimate the corresponding success probability. We do this by running $P^\star$ from each state many times, and counting how long it takes for $P^\star$ to complete $|y|$ proofs starting from that state. We interleave the counting for all of the states corresponding to the j-th proof to ensure that we can stop as soon as one has resulted in $|y|$ successful proofs. (This allows us to avoid running for too long just because one candidate state was bad.) Finally, we consider the best states for $j = 1, \ldots, \ell$, select the one with the highest estimated success probability, and use it as described above.

Our extractor. To summarize, our extractor works as follows:

1. For each $j = 1, \ldots, \ell$:
   (a) Collect $n = |y|$ candidate states, where $P^\star$ has successfully completed the first $j-1$ proofs and has a reasonable chance of successfully completing the next one.
   (b) Repeatedly run the next UA from all the above n states, and identify the state $\mathit{beststate}_j$ that is the first one that reaches n accepting executions

[4] There is some subtlety here in that we must guarantee that we can always recognize which set of extracted bits is the correct one (the one which is consistent with some valid witness). To do this we make use of the prover's initial commitment to the witness - if the extracted bits are consistent with the initial commitment, then we assume that they are the correct ones.

[5] In fact this is slightly inaccurate, as we also need to guarantee, even when we run this extractor many times and boost $P^\star$'s success probability a bit more, none of these times takes too long. We do this by estimating a reasonable upper bound for the running time of E based on our estimate of P^*'s success probability, and stopping E early if it runs more than this number of steps. As a result, we have to run E a few more times, and we set $\ell = d + 2$.

of the next $\langle P_{UA}(\cdot,\cdot), V_{UA}(\cdot)\rangle$. Let m_j be the number of such executions from that state.

2. Given the above m_j for $j = 1, \ldots, \ell$, let $\hat{j}$ be the index of the UA where $m_{\hat{j}}$ is minimal. $\frac{n}{2m_{\hat{j}}}$ is an estimate of a lower bound on the adversarial prover's success probability $p_{\hat{j}}$ in state *beststate*$_{\hat{j}}$.
3. Run approximately $q(|y|)/p_{\hat{j}}$ instances of the UA extractor. Return the result that agrees with the initial commitment sent by the prover.

A more formal description of the extractor and a more detailed analysis can be found in the full version of the paper. We also note that for technical reasons, we also need a statistically hiding trapdoor commitment scheme and a statistical zero-knowledge argument of knowledge of the trapdoor, which help us to prove the witness indistinguishability of our construction.

Theorem 2. *Under the assumption that a family of collision-resistant hash functions exists, then the protocol depicted in Fig. 1 is a constant-round witness indistinguishable UAQK.*

3 Secure Database Commitments

We consider the notion of a secure database commitment such that each element x appears at most once. As in [14], we will consider a formulation that captures both sets and databases. For a set S, we can define $Db[x]$ to be 1 if x appears in the set and $\perp$ if it does not. More generally, for a database of pairs (x, y), we define $Db[x]$ to be y if x appears in the set, and y otherwise. Since the difference between the two primitives is almost cosmetic, we will use the terms of sets and database interchangeably. We will assume that each element x belongs to $\{0,1\}^L$ and L is polynomial in the security parameter k. Thus, by the requirements above, this means that the database contains at most 2^L entries. We will make no other requirement on the size of the database.

The functionality in question is $\mathcal{F}_{DbCom}$ and is the natural extension of the standard commitment functionality, but in $\mathcal{F}_{DbCom}$ we must hide the size of the committed message and we also have to deal with queries and responses. (These responses must hide all other information about the database, thus there are also similarities with the zero-knowledge functionality.)

Ideal functionality $\mathcal{F}_{DbCom}$. The database commitment functionality consists of two phases: one in which a prover commits to a database, and a second in which the prover answers queries about that database. Here we will generalize this definition to say that the prover can commit to any database for which he knows an implicit representation. In particular, we will allow the prover to commit to a circuit which evaluates the desired database: it takes as input a string x, and returns $Db[x]$ (which will be $\perp$ if x is not in the database).

The formal specification is given in Fig. 2. For simplicity we assume that all queries made by the honest verifier V are fixed in advance. The definition can be generalized to adaptive queries, where the next query depends on the output of the previous ones. Our construction will satisfy this stronger definition as well.

Remark. Note that our definition of $\mathcal{F}_{DbCom}$ does not exclude the possibility that a player is able to commit to a very large (e.g. superpolynomial sized) set, for which he only knows some implicit representation. However, this would not violate any intuitive security requirement of secure database commitments since the critical requirement is that the prover is committed to some database (whose size is not a priori upperbounded by any fixed polynomial) at the end of the commitment phase, and that it must answer correctly according to this database during the query phase. We will show a construction in Section 4 that will require that the honest sender knows an explicit representation of the database he is committing to (i.e., the honest sender runs on input the sequence of elements it wants to commit to) (Note that an explicit representation can always be efficiently converted into an implicit representation $\mathcal{C}_{Db}$.) Obtaining a scheme where a real-world honest prover is allowed to have as input a polynomial-sized implicit representation that corresponds to a super-polynomial number of elements is also an interesting direction, and we defer it to future research.

Defining security. In the ideal execution, when initiated with input, parties interact with $\mathcal{F}_{DbCom}$. The adversary Sim has control of a party and thus can deviate from the prescribed computation, while the other player H_P follows the prescribed computation. We denote by $IDEAL^{\mathcal{F}_{DbCom}}_{H_P,\mathsf{Sim}}(k,z)$ the output of Sim on input an auxiliary input z, security parameter k and uniform randomness. Moreover we denote by $\{IDEAL^{\mathcal{F}_{DbCom}}_{H_P,\mathsf{Sim}}(k,z)\}$ the corresponding ensemble of distributions, with $k \in \mathcal{N}$ and $z \in \{0,1\}^*$.

In the real execution parties run a given protocol π. Let H_P be the honest party and $\mathcal{A}$ be the adversary, which has control of the other party. We denote by $EXEC^{\pi}_{H_P,\mathcal{A}}(k,z)$ the output of $\mathcal{A}$ on input an auxiliary input z, security parameter k and uniform randomness. Moreover we denote by $\{EXEC^{\pi}_{H_P,\mathcal{A}}(k,z)\}$ the corresponding ensemble of distributions, with $k \in \mathcal{N}$ and $z \in \{0,1\}^*$.

Definition 2. *A protocol π securely realizes the functionality $\mathcal{F}_{DbCom}$ if for any real-world adversary $\mathcal{A}$ there exists an ideal-world adversary Sim such that for every sufficiently large k, and for every $z \in \{0,1\}^*$, $\{EXEC^{\pi}_{H_P,\mathcal{A}}(k,z)\}$ and $\{IDEAL^{\mathcal{F}_{DbCom}}_{H_P,\mathsf{Sim}}(k,z)\}$ are computationally indistinguishable.*

4 Constant-Round Secure Database Commitments

In this section we present a constant-round protocol that securely realizes the $\mathcal{F}_{DbCom}$ functionality. Our construction uses a constant-round zero-knowledge argument of knowledge (ZKAoK) for all $\mathtt{NP}$, a constant-round WI universal argument of quasi knowledge, a 2-round trapdoor commitment scheme, and a zero-knowledge set scheme with two additional properties. As all of these ingredients can be instantiated through CRHFs, this gives a secure realization of $\mathcal{F}_{DbCom}$ based only on CRHFs. An additional feature of our protocol is that the amortized round complexity of a proof is optimal (i.e., proofs are actually non-interactive).

Functionality $\mathcal{F}_{DbCom}$

Players: Prover P, Verifier V.
Input of P: circuit $\mathcal{C}_{Db}$. **Input**[a] **of V:** $x_1, \ldots, x_m$ with $m = \mathsf{poly}(k)$.
Computation of $\mathcal{F}_{DbCom}$:

1. Upon receiving ($\texttt{Commit}, \mathcal{C}_{Db}$) from P, if ($\texttt{Commit}, \mathcal{C}_{Db}$) was already received then ignore, otherwise record ($\mathcal{C}_{Db}$), send ($\texttt{Receipt}$) to V.
2. Upon receiving ($\texttt{Query}, x$) from V, if ($\texttt{Commit}, \mathcal{C}_{Db}$) was previously received from P, then send ($\texttt{Query}, x$) to P, otherwise ignore.
3. Upon receiving ($\texttt{Open}, 1, x$) from P, if ($\texttt{Query}, x$) was not previously sent to P then ignore, otherwise evaluate the circuit $\mathcal{C}_{Db}$ on input x to obtain output $Db[x]$, and send ($\texttt{Open}, x, Db[x]$) to V.
4. Upon receiving ($\texttt{Open}, 0, x$) from P, if ($\texttt{Query}, x$) was not previously sent to P then ignore, otherwise send ($\texttt{Open-Abort}, x$) to V.
5. Upon receiving ($\texttt{Halt}$) from V, if the same message was previously received from V then ignore, otherwise send ($\texttt{Halt}$) to P.

Computation of P:

1. Upon activation, send ($\texttt{Commit}, \mathcal{C}_{Db}$) to $\mathcal{F}_{DbCom}$.
2. Upon receiving ($\texttt{Query}, x$) from $\mathcal{F}_{DbCom}$, send ($\texttt{Open}, 1, x$) to $\mathcal{F}_{DbCom}$.
3. Upon receiving ($\texttt{Halt}$) from $\mathcal{F}_{DbCom}$ go to Output.

Computation of V:

1. Upon receiving ($\texttt{Receipt}$) from $\mathcal{F}_{DbCom}$ send ($\texttt{Query}, x_1$) to $\mathcal{F}_{DbCom}$.
2. Upon receiving ($\texttt{Open}, x_i, y_i$) from $\mathcal{F}_{DbCom}$, if $0 < i < m$ then send ($\texttt{Query}, x_{i+1}$) to $\mathcal{F}_{DbCom}$.
3. Upon receiving ($\texttt{Open}, x_m, y_m$) or ($\texttt{Open-Abort}, x$) from $\mathcal{F}_{DbCom}$, send ($\texttt{Halt}$) to $\mathcal{F}_{DbCom}$ and go to Output.

Output:

P: Output $(x_1, \ldots, x_t)$ where $t = \mathit{poly}(k)$ and x_i for $0 < i \leq t$ is such that ($\texttt{Query}, x_i$) is the i-th message received from $\mathcal{F}_{DbCom}$.

V: Output $((x_1, y_1), \ldots, (x_{m'}, y_{m'}))$, where each x_i for $0 < i \leq m'$ was part of a message ($\texttt{Query}, x_i$) sent to $\mathcal{F}_{DbCom}$, and y_i for $0 < i \leq m'$ was part of a message ($\texttt{Open}, x_i, y_i$) received from $\mathcal{F}_{DbCom}$ while y_i is the empty string in case ($\texttt{Open-Abort}, x_i$) has been received from $\mathcal{F}_{DbCom}$.

[a] We stress that inputs can also be computed adaptively.

Fig. 2. Ideal computation of functionality $\mathcal{F}_{DbCom}$.

Zero-knowledge sets. We start by reviewing a major building block for our construction: zero-knowledge sets. At a high level, a non-interactive zero-knowledge set scheme is composed of four algorithms as follows: an algorithm $\mathsf{ZKSSetup}$, which generates some parameters, an algorithm ZKSCom, which commits to a database in a size independent way, an algorithm $\mathsf{ZKSProve}$, which allows the owner of the database to prove that a particular query response was consistent with the committed database, and an algorithm $\mathsf{ZKSVerify}$ for verifying such proofs. We will also use "ZKS proof system" to refer to the interaction between $\mathsf{ZKSProve}$ and $\mathsf{ZKSVerify}$.

We will use non-interactive zero-knowledge sets as a building block in our construction of secure database commitment. In the full version of this work we give a definition for zero-knowledge sets which is similar to that in [7]. We also define a somewhat stronger hiding property (which we call *special zero knowledge*), in which zero knowledge holds for all valid parameters output by the simulator set-up algorithm, and the simulated commitment is an honest commitment to an empty database. In the full version we will discuss how to achieve these properties based on CRHFs. We finally stress that even though non-interactive zero-knowledge sets have been defined in common reference string model, we will use them in the standard model by using interaction.

Intuition for our protocol. The basic idea behind this construction is fairly straightforward: we give a concise commitment to the database (using the ZKS commitment algorithm), and use a universal argument of quasi knowledge to prove knowledge of a valid opening. Then we can use the ZKS proof system to respond to queries. However, we need a few additional properties from the ZKS proof system to make this work. The issue is that when we are dealing with a corrupt prover, we need to be able to extract a circuit representation of the committed database from the UAQK (by which we mean a circuit which on input x returns $Db[x]$). On the other hand, the UAQK only guarantees an extractor which, given i, returns the ith bit of a valid witness. So we augment the definition of ZKS with an additional property which will allow us to construct an implicit database representation from a special witness string. In particular, we need to guarantee that the witness will be in a format such that we can efficiently extract $Db[x]$ for any x given only the UAQK extractor which allows queries to individual bits of the witness. Furthermore, in order to argue that the responses in the query phase must be consistent with the extracted database, we also need to be able to efficiently extract a ZKS proof for each x. This must still be done by just querying the circuit representation. Therefore the mere use of UAQK is not sufficient to securely realize $\mathcal{F}_{DbCom}$, and we will provide the needed additional ingredients.

Definition 3. *A ZKS proof system* ($\mathsf{ZKSSetup}, \mathsf{ZKSCom}, \mathsf{ZKSProve}, \mathsf{ZKSVerify}$) allows local witnesses *if there exist additional polynomial time deterministic algorithms* $\mathtt{FormWitness}, \mathsf{TMVer}, \mathsf{Eval}, \mathsf{PfGen}$ *such that the following properties hold:*

Witness verifiability. For any sufficiently large k, for all polynomial sized Db, and random strings r,

$$\Pr[\mathtt{ZKSPAR} \leftarrow \mathsf{ZKSSetup}(1^k); zks \leftarrow \mathsf{ZKSCom}(\mathtt{ZKSPAR}, Db, r);$$
$$w \leftarrow \mathtt{FormWitness}(\mathtt{ZKSPAR}, Db, r) : \mathsf{TMVer}(\mathtt{ZKSPAR}, zks, w) = 1] = 1.$$

Local evaluation. There exists a polynomial q such that for any sufficiently large k, for any $x \in \{0,1\}^k$ and $w \in \{0,1\}^{2^k}$, $\mathsf{Eval}(x, w)$ *only accesses $q(k)$ bits of w and runs in time polynomial in k.*

Local verification. There exists a polynomial q' such that for any sufficiently large k, $x \in \{0,1\}^k$ and $w \in \{0,1\}^{2^k}$, $\mathsf{PfGen}(x, w)$ *only accesses $q'(k)$ bits of w and runs in time polynomial in k. Moreover for any $x \in \{0,1\}^k$ and $w \in \{0,1\}^{2^k}$ and polynomial sized zks,*

$$\Pr[\mathtt{ZKSPAR} \leftarrow \mathsf{ZKSSetup}(1^k) : \mathsf{TMVer}(\mathtt{ZKSPAR}, zks, w) = 1 \wedge \\ \wedge \mathsf{ZKSVerify}(\mathtt{ZKSPAR}, zks, x, \mathsf{Eval}(x, w), \mathsf{PfGen}(x, w)) = 0] = 0.$$

The intuition of the above definition is that there must exist a procedure FormWitness that transforms the witness while still preserving the content of the database (indeed this is verifiable through TMVer). Moreover one can extract the membership or non membership of an element and the corresponding proof by accessing only a polynomial number of bits of this witness, according to the algorithms Eval and PfGen. The reason why we allow w to be of superpolynomial size is that the adversary can bypass the procedure FormWitness and produce a circuit that implicitly represent a superpolynomial sized witness. The interesting property of our definition is that even in this case, Eval and PfGen will be efficient. We now show that standard constructions of ZKS ([14,7,5]) have this property.

Previous ZKS schemes allow local witnesses. We will use ideas from the ZKS construction of [14,7]. We summarize only the main properties we need.

The constructions work by building a tree. For each x in the database, we parse x as bits $b_1, \ldots, b_L$. Then at position $b_1, \ldots, b_L$ in the tree (where $b_i = 0$ denotes the left branch and $b_i = 1$ denotes the right branch at height i), we store a commitment v_x to the corresponding y (all commitments here use a special commitment scheme). We construct the tree from the bottom up. For every ancestor $x' = b_1, \ldots, b_i$ of such an x, we compute a commitment to a hash of the two children: $v_{x'} = \mathsf{Com}(h(v_{x'||0}, v_{x'||1}))$. If one of the children has not yet been specified (it is the root of a subtree with no leaves in the database), we set that child node to $\mathsf{Com}(\perp)$.

Then a proof for value $(x, y) \in Db$ can be formed by producing all sibling and ancestor nodes and openings for all ancestor commitments. A proof for value $x \notin Db$ is more complex, but it can be formed by first finding the root of the largest empty subtree containing x. The value of this root node (call it $x_\perp$) should be $\perp$. Then the proof will include all sibling and ancestor nodes of $x_\perp$, special openings of all ancestor commitments, and an additional value which can be constructed given x and the randomness used to form the commitment at $x_\perp$.

More formally we have that for every empty subtree, there exists a polynomial sized string *info*, such that for all leaves x in this subtree, $\mathsf{PfGen}(x, \mathit{info})$ produces output identical to ZKSProve. It is also possible to efficiently verify that *info* is formed correctly for a given subtree.

We now sketch the necessary algorithms as defined in Definition 3:

FormWitness : will first form the ZKS tree as described above. Then we will transform this into a single linear witness. For each nonempty, non-$\perp$ node x in the tree we will form an entry consisting of the value of v_x, the values of its children, the opening of the commitment v_x, and the positions of the children nodes in the tree. For each $\perp$ node, we form an entry with the commitment $v_\perp$ and the extra *info* necessary to open the commitment. The witness will begin with the bit position of the entry corresponding to the root commitment.

TMVer: will traverse the entire tree, and verify all commitments and hashes. This algorithm can easily be described by a polynomial sized TM.

Eval(x): will follow the path through the tree corresponding to x, beginning with the root and continuing until it reaches either $\perp$ or a leaf node with value y, and return the result.

PfGen: will follow the path through the tree corresponding to x until it reaches an empty subtree or a leaf. If it is a leaf, it will return all of the ancestor and sibling nodes and the openings, as described above. If it is an empty subtree, it reads the accompanying value *info*, runs $\mathsf{PfGen}(x, \mathit{info}) \rightarrow \pi$ and returns π and all the ancestor and sibling nodes and openings. In both cases the output for an honestly generated witness will be identical to the output of ZKSProve.

PfVer: will verify each of the hashes and commitments in the chain. If the final node is $\perp$, it will verify the accompanying proof π.

Our construction. We will use a constant-round zero-knowledge argument of knowledge (ZKAoK) $\langle \mathcal{P}(\cdot,\cdot), \mathcal{V}(\cdot) \rangle$, a constant-round WI UAQK $\langle \mathsf{UAP}(\cdot,\cdot), \mathsf{UAV}(\cdot) \rangle$, a 2-round trapdoor commitment scheme ($\mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TDec}, \mathsf{Ver}$), and a special ZKS ($\mathsf{ZKSSetup}, \mathsf{ZKSCom}, \mathsf{ZKSProve}, \mathsf{ZKSVerify}$) which allows local verification with zero knowledge simulator ($\mathsf{ZKSSimSetup}, \mathsf{ZKSSimCom}, \mathsf{ZKSSimProve}$). A description of our scheme is found in Fig. 3.

Security Parameter: k.
Input to P**:** $Db = ((x_1, y_1), \ldots, (x_s, y_s))$ where $s = \mathsf{poly}(k)$ and $x_i \in \{0,1\}^k$ for $0 < i \leq s$.
Input to V**:** $x'_1, \ldots, x'_m$, where $m = \mathsf{poly}(k)$ and $x'_i \in \{0,1\}^k$ for $0 < i \leq m$.
Commitment Phase:

1. $V \rightarrow P$: set $(\mathtt{ZKSPAR}, \mathtt{ZKSTRAP}) \leftarrow \mathsf{ZKSSimSetup}(1^k)$, $(\mathtt{crs}, \mathtt{aux}) \leftarrow \mathsf{TGen}(1^k)$ and send $(\mathtt{ZKSPAR}, \mathtt{crs})$ to P.

2. $V \leftrightarrow P$: V proves knowledge of $\mathtt{ZKSTRAP}, \mathtt{aux}$ (i.e., V and P run $\mathcal{P}((\mathtt{ZKSPAR}, \mathtt{crs}), (\mathtt{ZKSTRAP}, \mathtt{aux}))$ and $\mathcal{V}((\mathtt{ZKSPAR}, \mathtt{crs}))$ respectively). If P rejects the proof, then it aborts.

3. $P \rightarrow V$: pick $r \in \{0,1\}^k$, set $(c, \mathsf{dec}) = \mathsf{Com}(\mathtt{crs}, zks = \mathsf{ZKSCom}(\mathtt{ZKSPAR}, Db, r))$ and send c to V.

4. $P \leftrightarrow V$: P execute $\mathtt{FormWitness}(\mathtt{ZKSPAR}, Db, r)$ to generate witness w, and then execute the code of UAP on input $c||\mathtt{ZKSPAR}||\mathtt{crs}$ with witness $zks||\mathsf{dec}||w$. V executes the code of UAV on input $c||\mathtt{ZKSPAR}||\mathtt{crs}$. If UAV rejects, V aborts. UAP proves quasi knowledge of a witness of the form $zks||\mathsf{dec}||w$, where zks, dec is a valid opening for commitment c under $\mathtt{crs}$, and w is such that $\mathsf{TMVer}(\mathtt{ZKSPAR}, zks, w)$ accepts.

5. $P \rightarrow V$: open the commitment c by sending zks, dec to V. If the opening is not correct, V aborts.

Query/Answer Phase:
For $i = 1, \ldots, m$ do:

6. $V \rightarrow P$: send x'_i to P.

7. $P \rightarrow V$: send $(y'_i = Db[x'_i], \pi = \mathsf{ZKSProve}(\mathtt{ZKSPAR}, x'_i, Db[x'_i], Db, r))$ to V. V outputs (x'_i, y'_i) if $\mathsf{ZKSVerify}(\mathtt{ZKSPAR}, zks, x'_i, y'_i, \pi) = 1$ and $(x'_i, \perp)$ otherwise.

Fig. 3. Our scheme for secure database commitments.

Round optimality. We stress that simply by inspection one can observe that the query/proof phase is non-interactive (optimal).

Security. The protocol depicted in Fig. 3 securely realizes the $\mathcal{F}_{DbCom}$ functionality, i.e., it is a constant-round protocol for secure database commitment. For lack of space the proof is presented in the full version.

Theorem 3. *If* $\langle \mathcal{P}(\cdot,\cdot), \mathcal{V}(\cdot)\rangle$ *is a ZKAoK,* $\langle \mathsf{UAP}(\cdot,\cdot), \mathsf{UAV}(\cdot)\rangle$ *is a WI UAQK,* $(\mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TDec}, \mathsf{Ver})$ *is a 2-round trapdoor commitment scheme, and* $(\mathsf{ZKSSetup}, \mathsf{ZKSCom}, \mathsf{ZKSProve}, \mathsf{ZKSVerify})$ *is a special zero-knowledge set scheme which allows local witnesses, then the protocol depicted in Fig. 3 securely realizes the* $\mathcal{F}_{DbCom}$ *functionality.*

Corollary 1. *Under the assumption that there exists a family of CRHFs, then there exists an efficient protocol which securely realizes the* $\mathcal{F}_{DbCom}$ *functionality.* This follows from the fact that all of the primitives mentioned in Theorem 3 can be realized based on CRHFs.

References

1. Ateniese, G., De Cristofaro, E., Tsudik, G.: (If) size matters: Size-hiding private set intersection. In: PKC 2011. pp. 156–173. Springer.
2. Barak, B.: How to Go Beyond the Black-Box Simulation Barrier. In: FOCS 2001. pp. 106–115. IEEE Computer Society Press.
3. Barak, B.: Non-Black-Box Techniques in Cryptography, Ph.D. Thesis. Weizmann Institute of Science (2004)
4. Barak, B., Goldreich, O.: Universal Arguments and Their Applications. In: CCC 2002. IEEE Computer Society Press.
5. Catalano, D., Dodis, Y., Visconti, I.: Mercurial commitments: Minimal assumptions and efficient constructions. In: TCC 2006. pp. 120–144. Springer.
6. Catalano, D., Fiore, D., Messina, M.: Zero-knowledge sets with short proofs. In: Eurocrypt 2008. pp. 433–450. Springer.
7. Chase, M., Healy, A., Lysyanskaya, A., Malkin, T., Reyzin, L.: Mercurial commitments with applications to zero-knowledge sets. In: Eurocrypt 2005. pp. 422–439. Springer-Verlag.
8. Gennaro, R., Micali, S.: Independent zero-knowledge sets. In: ICALP 2006. pp. 181–234. Springer.
9. Goldreich, O.: Foundations of Cryptography - Volume II - Basic Applications. Cambridge press (2004)
10. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game - A Completeness Theorem for Protocols with Honest Majority. In: STOC 1987. pp. 218–229
11. Hada, S., Tanaka, T.: On the existence of 3-Round zero-knowledge protocols. In: Crypto 1998.
12. Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: TCC 2007. pp. 575–594. Springer.
13. Libert, B., Yung, M.: Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In: TCC 2010 pp. 499–517. Springer.
14. Micali, S., Rabin, M., Kilian, J.: Zero-knowledge sets. In: FOCS 2003. pp. 80–91.
15. Prabhakaran, M., Xue, R.: Statistically hiding sets. In: CT-RSA 2009. pp. 100–116. Springer.