Statistically Hiding Sets

Manoj Prabhakaran
Dept. of Computer Science
University of Illinois, Urbana-Champaign
e-mail: mmp@cs.uiuc.edu

Rui Xue
State Key Laboratory of Information Security
Institute of Software, Chinese Academy of Sciences
e-mail: rxue@is.iscas.ac.cn

Abstract

Zero-knowledge set is a primitive introduced by Micali, Rabin, and Kilian (FOCS 2003) which enables a prover to commit a set to a verifier, without revealing even the size of the set. Later the prover can give zero-knowledge proofs to convince the verifier of membership/non-membership of elements in/not in the committed set.

We present a new primitive called *Statistically Hiding Sets* (SHS), similar to zero-knowledge sets, but providing an information theoretic hiding guarantee. This is comparable to relaxing zero-knowledge proofs to *witness independent proofs*. More precisely, we continue to use the simulation paradigm for our definition, but do not require the simulator (nor the distinguisher) to be efficient.

We present a new scheme for statistically hiding sets, which does not fit into the "Merkle-tree/mercurial-commitment" paradigm used for *all* zero-knowledge set constructions so far. This not only provides some efficiency gains compared to the best possible schemes in that paradigm, but also lets us provide *statistical* hiding, without the prover having to maintain growing amounts of state with each new proof; this is not known to be possible with the previous approach.

Our construction is based on an algebraic tool called *trapdoor DDH groups* (TDG), introduced recently by Dent and Galbraith (ANTS 2006). Ours is perhaps the first non-trivial application of this tool. However the specific hardness assumptions we associate with TDG are different, and of a strong nature — strong RSA and a knowledge-of-exponent assumption. Our new knowledge-of-exponent assumption may be of independent interest.

Keywords: Statistically Hiding Set, Zero-Knowledge Set, Trapdoor DDH group, Statistical Zero-Knowledge, Accumulator, Knowledge-of-Exponent Assumption.

Contents

1 Introduction

Zero-knowledge set is a fascinating cryptographic primitive introduced by Micali, Rabin, and Kilian [23], which has generated much interest since then [25, 22, 10, 9, 17]. It enables a party (the prover) to commit a set – without revealing even its size – to another party (the verifier). Later the verifier can make membership queries with respect to the committed set; the prover can answer these queries and give proofs to convince the verifier of the correctness of the answers, without revealing anything further about the set.

In this paper we present a novel scheme for zero-knowledge sets (but defined differently — see below), significantly departing from previous approaches. While *all* previous approaches used a tree-based construction (along with a primitive called mercurial commitments), we present a direct algebraic construction. To the best of our knowledge, this is the first construction of zero-knowledge sets that does not fit into the Merkle-tree/mercurial-commitment paradigm. This not only provides some efficiency gains, but also opens up the possibility of constructing schemes with additional properties which are not amenable to the Merkle-tree based construction. For instance, our construction provides *statistical* zero-knowledge, without the prover having to maintain growing amounts of state with each new proof; this is not known to be possible with the previous approach.

Our definition of zero-knowledge sets is different from the original one by Micali et al. [23]. This new definition is another contribution of this work. For clarity, in the rest of the paper we use the term *statistically hiding sets* (SHS) for our modified definition. On the one hand it is stronger, in that it requires security even against computationally unbounded corrupt verifiers, while on the other hand it is weaker in that it provides a witness-independence like guarantee (or alternately, an unbounded simulation based zero-knowledge guarantee) instead of a traditional zero-knowledge (with efficient simulation) guarantee. Further, we define SHS in the classical two-party setting of a prover and a verifier, *without any trusted setup*, whereas [23] allows a common reference string, but requires the proofs to be verifiable by multiple verifiers. We discuss the differences in the definitions in more detail below, in Section 1.1.

We offer SHS as a (comparable) alternative to ZK sets. The intuitive security guarantees provided by SHS is the same as that provided by ZK sets. (In particular, the informal description in the opening paragraph of this section is applicable to both.) But the technical differences help us in providing a construction which is fundamentally different from previous approaches.

As already mentioned, our techniques are algebraic. It is based on *trapdoor DDH groups* (TDG), a primitive introduced recently by Dent and Galbraith [13]. Ours is perhaps the first non-trivial application of this cryptographic primitive. However the specific hardness assumptions we associate with TDG are different, and of a strong nature (strong RSA and a Knowledge-of-Exponent assumption). While we believe these assumptions are reasonable given the state-of-the-art in algorithmic number theory, we do hope that our approach leads to newer efficient constructions of statistically hiding sets and similar tools *based on more standard assumptions*. We also hope that our work will help in better understanding certain powerful assumptions. In particular, the new simple and powerful knowledge-of-exponent assumption that we introduce (and prove to hold in a generic group model) may be of independent interest. See Section 1.2 below for more details.

Related Work. Micali, Rabin and Kilian introduced the concept of zero-knowledge sets, and provided a construction based on a Merkle-tree based approach [23]. All subsequent constructions have followed this essential idea, until now. Chase et al. abstracted the properties of the commit-

ments used in [23] and successfully formalized a general notion of commitments named *mercurial commitments* [10]. Catalano et al. [9] further clarified the notion of mercurial commitments used in these constructions.

Liskov [22] augmented the original construction of Micali et al. to be updatable. Gennaro and Micali [17] introduced a non-malleability-like requirement called independence and constructed mercurial commitments with appropriate properties which when used in the Merkle-tree based construction resulted in a zero-knowledge set scheme with the independence property. Ostrovsky et al. [25] extended the Merkle-tree based approach to handle more general datastructures (directed acyclic graphs); however in their notion of privacy the size of the data-structure is allowed to be publicly known, and as such they do not require mercurial commitments.

Our construction is motivated by prior constructions of *accumulators*. The notion of accumulators was first presented in [6] to allow rolling a set of values into one value, such that there is a short proof for each value that went into it. Barić and Pfitzmann [4] proposed a construction of a collision-resistant accumulator under the strong RSA assumption. Camenisch and Lysyanskaya [7] further developed a dynamic accumulator so that accumulated values can be added or removed from the accumulator. The most important difference between an accumulator and a zero-knowledge set is that the former does not require the prover to provide a proof of non-membership for elements not accumulated.

Trapdoor DDH groups were introduced by [13], and to the best of our knowledge has not been employed in any cryptographic application (except a simple illustrative example in [13]). They also gave a candidate for this primitive based on elliptic curve groups with composite order, using "hidden pairings." Indeed, Galbraith and McKee [15] had pointed out that if the pairing operation is not hidden, typical hardness assumptions like the RSA assumption may not be justified in those groups. [13] also defined another primitive called Trapdoor Discrete Logarithm groups; however the candidate they proposed for this – with some reservations – was subsequently shown to be insecure [24].

Our hardness assumptions on TDG are different from those in [13]. The first assumption we use, namely the Strong RSA assumption, was introduced by Barić and Pfitzmann in their work on accumulators [4] mentioned above, as well as by Fujisaki and Okamoto [14]. Subsequently it has been studied extensively and used in a variety of cryptographic applications (for e.g. [16, 11, 8, 2, 7, 3, 26]). The second assumption we make is a Knowledge-of-Exponent Assumption (KEA). The first KEA, now called KEA1, was introduced by [12], to construct an efficient public key cryptosystem secure against chosen ciphertext attacks. Hada and Tanaka [20] employed it together with another assumption (called KEA2) to propose 3-round, negligible-error zero-knowledge arguments for NP. Bellare and Palacio in [5] falsified the KEA2 assumption and used another extension of KEA1, called KEA3, to restore the results in [20].

1.1 Differences with the original definition

Our definition differs from that of Micali, Rabin and Kilian [23] in several technical aspects. The original definition was in the setting of the trusted setup of a common reference string; it also required proofs to be publicly verifiable; the zero-knowledge property was required for PPT verifiers and was defined in terms of a PPT simulator (the indistinguishability could be computational, statistical or perfect).

In contrast, we define an SHS scheme as a two-party protocol, with no requirement of public verifiability of the proofs; but we do not allow a trusted setup. We require the hiding property

to hold even when the verifier is computationally unbounded; but we do not require an efficient simulation.

As mentioned before, we consider these differences to be of a technical nature: the basic utility provided by SHS is the same as that by ZK sets. However we do expect qualitative differences to show up when considering composition issues (cf. parallel composition of zero-knowledge proofs and witness indistinguishable proofs). We leave this for future investigation.

Our definition of the statistical hiding property is formulated using a computationally unbounded simulation. It is instructive to cast our security definition (when the verifier is corrupt) in the "real-world/ideal-world" paradigm of definitions that are conventional for multi-party computation. In the ideal world the corrupt verifier (simulator) gets access only to a blackbox to answer the membership queries. We require a statistically indistinguishable simulation — effectively allowing a computationally unbounded "environment." (However our definition is not in the Universal Composition framework, as we do not allow the environment to interact with the adversary during the protocol.)

Remark: Computationally Unbounded Verifiers. Our definition of security is meaningful and non-trivial even for a (real) corrupt verifier that is computationally unbounded. (That is, not only the simulator, but also the adversary can be computationally unbounded.) This is in stark contrast with the setting of zero-knowledge *proofs*, wherein if the corrupt verifiers are computationally unbounded, the notion becomes trivial.

To demystify this difference, we point out the obvious, but important conceptual difference between zero-knowledge sets (or SHS) and zero-knowledge proofs: in zero-knowledge proofs the statement being proven is about a property of a common input (i.e., membership of a common input in a language), that can in fact be computed given sufficient computational power. In contrast, the statements proven in a zero-knowledge set (or SHS) scheme relate to hidden *information*, uncertainty about which remains no matter how much computational power is available.

If considering computationally unbounded corrupt verifiers, the simulator *needs to be* unbounded and the zero-knowledge property has to be statistical (as is the case in SHS).

Remark: Minimal computational restriction for a meaningful definition. The classical (stand-alone) definition of zero-knowledge proofs exhibits an interesting duality with our definition of statistically-hiding sets. There, security is required against an unbounded prover and a computationally bounded verifier. As mentioned above, bounding the verifier is the minimum restriction that is required to make the ZK proof notion *non-trivial*. For us, bounding the prover is the minimum restriction required to make the SHS notion *realizable* at all. (An unbounded prover can find multiple sets to explain a single commitment; note that the size of the commitment cannot grow with the size of the set, and hence it must necessarily have collisions.)

Remark: Absence of Trusted Setup. In the case of the original zero-knowledge sets and non-interactive zero-knowledge proofs (NIZKs) a trusted setup like a common reference string is essential. It is not possible to allow the verifier or the prover to carry out this setup. In contrast, our definition of SHS requires the verifier to carry out the setup (and indeed, this is the case in the construction we present). The private information generated during the setup may later be used by the verifier to efficiently verify the (non-interactive) proofs. To enable such use of private information we remove the requirement of public verifiability. (However, we remark that even if we

allow a trusted setup, the construction we present does not allow public verifiability. But, in our construction, the proofs do remain publicly verifiable for a *computationally unbounded* verifier.)

1.2 Assumptions Used

The hardness asssumptions used in this work are of a strong nature. We use a combination of a strong RSA assumption and a knowledge-of-exponent assumption. Further these assumptions are applied to a relatively new family of groups, namely, trapdoor DDH groups. Therefore we advise caution in using our protocol before gaining further confidence in these assumptions. Nevertheless we point out that *the assumptions are used only for soundness*. The statistical hiding property is *unconditional*. This means that even if an adversary manages to violate our assumptions *after* one finishes using the scheme, it cannot violate the security guarantees at that point.

The knowledge of exponent assumption we use — called KEA-DH — is a new proposal, but similar to KEA1 introduced in 1991 by Damgard [12]. We believe this powerful assumption could prove very useful in constructing efficient cryptographic schemes, yet is reasonable enough to be safely assumed in different families of groups. In this work we combine KEA-DH with another (more standard) assumption called the strong RSA assumption. In Section 3 we descibe these assumptions, and in Appendix B we provide some further preliminary observations about them (including a proof that KEA-DH holds in a generic group model).

Our construction depends on the idea of trapdoor DDH groups by Dent and Galbraith [13].[1] But our use of this primitive does not require exactly the same features as a trapdoor DDH scheme offers.[2] We believe this is a tool that has the potential for realizing further interesting cryptographic concepts. In Appendix B.1 we illustrate the plausibility of the assumptions by showing that the KEA-DH assumption holds in the generic group model (even if endowed with a bilinear operation and a DDH oracle). Nevertheless, we caution the reader that the candidate proposed in [13] is relatively new, and calls for further investigation.

2 Statistically Hiding Sets

In this section we present our definition of statistically hiding sets (SHS). It is slightly different from the original definition of zero-knowledge sets of Micali et al. [23], but offers the same intuitive guarantees.

Notation. We write Pr[experiment : condition] to denote the probability that a condition holds after an experiment. An experiment is a probabilistic computation (typically described as a sequence of simpler computations); the condition will be in terms of variables set during the experiment. We write {experiment : random variable} to denote the distribution in which a random variable will be distributed after an experiment.

[1] See Appendix B for a brief outline of the candidate proposed in [13].

[2] In particular, since we do not use the DDH assumption, it is conceivable that in an implementation, the "trapdoor" for the DDH problem could be known publicly and still the security assumptions we use would hold. (This would allow our construction to be publicly verifiable in the common reference string model.) But for the particular candidate TDDH group in [13], if the trapdoor is made public, even our assumptions do not hold any more. So we do not propose a stronger primitive (that keeps the KEA-DH and strong RSA assumptions but makes the DDH problem easy for all), as we would not have a candidate for it at this point.

The security parameter will be denoted by k. By a PPT algorithm we mean a non-uniform probabilistic algorithm which terminates within $\text{poly}(k)$ time for some polynomial poly. We say two distributions are *almost the same* if the statistical difference between them is negligible in k.

Statistically Hiding Set Scheme: Syntax. A (non-interactive) SHS scheme consists of four PPT algorithms: setup, commit, prove and verify.

- setup: This is run by the verifier to produce a public-key/private-key pair: $(\mathsf{PK}, \mathsf{VK}) \leftarrow \mathsf{setup}(1^k)$. Here k is the security parameter; the public-key PK is to be used by a prover who makes a commitment to the verifier, and the private (verification) key VK is used by the verifier to verify proofs of membership and non-membership.

- commit: This algorithm is used by the prover to generate a commitment σ of a finite set S, along with private information ρ used for generating proofs: it takes as input $(\sigma, \rho) \leftarrow \mathsf{commit}(S, \mathsf{PK})$.

- prove: This algorithm is used by the prover to compute non-interactive proofs of memberships or non-memberships of queried elements: $\pi \leftarrow \mathsf{prove}(x, \rho, \mathsf{PK})$. Here π is a proof of membership or non-membership of x in a set S that was committed using commit; ρ is the the private information computed by commit.

- verify: The verifier verifies a given proof of membership or non-membership using the algorithm verify: $b \leftarrow \mathsf{verify}(\pi, \sigma, x, \mathsf{VK})$. The output b is either a bit 0 or 1 (corresponding to accepting a proof of non-membership, or a proof of membership, respectively), or $\perp$ (corresponding to rejecting the proof).

Statistically Hiding Set Scheme: Security. Let U_k stand for the universal set of elements allowed by the scheme for security parameter k (i.e., the sets committed using the scheme are subsets of U_k). U_k can be finite (e.g., $\{0,1\}^k$) or infinite (e.g. $\{0,1\}^*$). We say that PPT algorithms $(\mathsf{setup}, \mathsf{commit}, \mathsf{prove}, \mathsf{verify})$ form a secure SHS scheme if they satisfy the properties stated below.

- *Perfect completeness:* For any finite set $S \subseteq U_k$, and $x \in U_k$,

$$\Pr\Big[(\mathsf{PK}, \mathsf{VK}) \leftarrow \mathsf{setup}(1^k); (\sigma, \rho) \leftarrow \mathsf{commit}(\mathsf{PK}, S); \\ \pi \leftarrow \mathsf{prove}(\mathsf{PK}, \rho, x); b \leftarrow \mathsf{verify}(\mathsf{VK}, \pi, \sigma, x) \\ : x \in S \implies b = 0 \text{ and } x \notin S \implies b = 1\Big] = 1$$

  That is, if the prover and verifier honestly follow the protocol, then the verifier is always convinced by the proofs given by the prover (it never outputs $\perp$).

- *Computational Soundness:* For any PPT adversary $\mathcal{A}$, there is a negligible function ν s.t.

$$\Pr\Big[(\mathsf{PK}, \mathsf{VK}) \leftarrow \mathsf{setup}(1^k); (\sigma, x, \pi_0, \pi_1) \leftarrow \mathcal{A}(\mathsf{PK}); \\ b_0 \leftarrow \mathsf{verify}(\mathsf{VK}, \pi_0, \sigma, x); b_1 \leftarrow \mathsf{verify}(\mathsf{VK}, \pi_1, \sigma, x) \\ : b_0 = 0 \text{ and } b_1 = 1\Big] \leq \nu(k)$$

  That is, except with negligible probability, an adversary cannot produce a commitment σ and two proofs π_0, π_1 which will be accepted by the verifier respectively as a proof of non-membership and of membership of an element x in the committed set.

- *Statistical Hiding:* There exist distributions $\mathsf{dummycommit}(\mathsf{PK})$, and $\mathsf{dummyprove}_0(\mathsf{PK}, \rho, x)$ and $\mathsf{dummyprove}_1(\mathsf{PK}, \rho, x)$, such that for every adversary $\mathcal{A}$ (not necessarily PPT), and every finite $S \subseteq U_k$ and any polynomial $t = t(k)$, the following two distributions have a negligible statistical distance between them.

$$\left\{\begin{array}{l}(\mathsf{PK}, s_0) \leftarrow \mathcal{A}(1^k);\\ (\sigma, \rho) \leftarrow \mathsf{commit}(\mathsf{PK}, S);\\ \pi_0 := \sigma;\\ \texttt{for } i = 1, \ldots, t\\ \quad (x_i, s_i) \leftarrow \mathcal{A}(s_{i-1}, \pi_{i-1});\\ \quad \pi_i \leftarrow \mathsf{prove}(\mathsf{PK}, \rho, x_i);\\ \texttt{endfor}\\ \quad : (\vec{s}, \sigma, \vec{\pi})\end{array}\right\} \quad \left\{\begin{array}{l}(\mathsf{PK}, s_0) \leftarrow \mathcal{A}(1^k);\\ (\sigma, \rho) \leftarrow \mathsf{dummycommit}(\mathsf{PK});\\ \pi_0 := \sigma;\\ \texttt{for } i = 1, \ldots, t\\ \quad (x_i, s_i) \leftarrow \mathcal{A}(s_{i-1}, \pi_{i-1});\\ \quad \pi_i \leftarrow \mathsf{dummyprove}_{\chi^S_{x_i}}(\mathsf{PK}, \rho, x_i);\\ \texttt{endfor}\\ \quad : (\vec{s}, \sigma, \vec{\pi})\end{array}\right\}$$

  Here χ^S_x is a bit indicating $x \in S$ or not. $\vec{\pi}$ stands for the list of proofs $(\pi_1, \ldots, \pi_t)$, and $\vec{s}$ for the output of the adversay $(s_0, \ldots, s_t)$.

  Note that the second distribution does not depend on S, but only on whether $x_i \in S$ for $i = 1, \ldots, t$. We *do not* require the distributions $\mathsf{dummycommit}(\cdot)$ and $\mathsf{dummyprove}(\cdot)$ to be efficiently sampleable.

  An alternate (but equivalent) definition of the hiding property is in terms of a (computationally unbounded) simulator which can statistically simulate an adversary's view, with access to only an oracle which tells it whether $x \in S$ for each x for which it has to produce a proof.

Note that we restrict ourselves to a "stand-alone" security definition. However, we do allow the verifier auxiliary information about the set S being committed to, by virtue of the order of quantifiers in the definition of the hiding property.

3 Trapdoor DDH Group

Following Dent and Galbraith[13], we define a *trapdoor DDH group* (TDG, for short) as follows. The hardness assumptions we make on such a group are different.

A TDG is defined by two PPT algorithms: a group generation algorithm Gen and a trapdoor DDH algorithm TDDH.

- Gen takes as input the security parameter k (in unary), and outputs a description (i.e., an algorithm for the group operation) for a group $\mathbb{G}$, an element $g \in \mathbb{G}$, an integer $n = 2^{O(k)}$ as an upperbound on the order of g in $\mathbb{G}$, and a trapdoor τ. The representation of this output should be such that given a tuple $(\mathbb{G}, g, n)$ (purportedly) produced by $\mathsf{Gen}(1^k)$, it should be possible to efficiently verify that $\mathbb{G}$ is indeed a group and $g \in \mathbb{G}$ has order at most n.[3]

[3]The upperbound on the order of g can often be the order of $\mathbb{G}$ itself, which could be part of the description of $\mathbb{G}$. We include this upperbound explicitly in the output of Gen as it plays an important role in the construction and the proof of security.

- TDDH takes as input $(\mathbb{G}, g, n, \tau)$ produced by Gen, as well as elements A, B, C and outputs 1 if and only if $A, B, C \in \mathbb{G}$ and there exist integers a, b, c such that $A = g^a, B = g^b, C = g^{ab}$.

We make the following two hardness assumptions on a TDG.

Assumption 1 (Strong RSA Assumption) *For every PPT algorithm $\mathcal{A}$, there is a negligible function ν such that*

$$\Pr\Big[(\mathbb{G}, g, \tau) \leftarrow \mathsf{Gen}(1^k); \\ (x, e) \leftarrow \mathcal{A}(\mathbb{G}, g) \\ : x^e = g, e > 1\Big] < \nu(k).$$

The probability is over the coin tosses of Gen *and* $\mathcal{A}$.

Assumption 2 (Diffie-Hellman Knowledge of Exponent Assumption (KEA-DH)) *For every* PPT *adversary $\mathcal{A}$, there is a* PPT *extractor $\mathcal{E}$ and a negligible function ν such that*

$$\Pr\Big[(\mathbb{G}, g, \tau) \leftarrow \mathsf{Gen}(1^k); \\ (A, B, C) \leftarrow \mathcal{A}(\mathbb{G}, g; r); \\ z \leftarrow \mathcal{E}(\mathbb{G}, g; r) \\ : \left(\exists a, b : A = g^a, B = g^b, C = g^{ab}\right), C \neq A^z, C \neq B^z\Big] < \nu(k).$$

Here r stands for the coin tosses of $\mathcal{A}$; $\mathcal{E}$ may toss additional coins. The probability is over the coin tosses of Gen*, $\mathcal{A}$ (namely r), and any extra coins used by $\mathcal{E}$.*

Informally, KEA-DH says that if an adversary takes g generated by Gen and outputs a DDH tuple (g^a, g^b, g^{ab}), then it must know either a or b. However, since $\mathcal{A}$ may not know the order of g in $\mathbb{G}$, the integers a, b are not unique.

For our use later, we remark that the extractor $\mathcal{E}$ can be modified to output its inputs as well as the output of $\mathcal{A}$, along with z. Then KEA-DH asserts that if there exists an adversary that takes $(\mathbb{G}, g)$ as input and produces (α, A, B, C) which satisfies some prescribed property and in which (A, B, C) is a DDH tuple, with non-negligible probability, then there is a machine $\mathcal{E}$ which takes $(\mathbb{G}, g)$ as input, and with non-negligible probability outputs $(\mathbb{G}, g, \alpha, A, B, C, z)$ such that (α, A, B, C) satisfies the prescribed property and either $C = A^z$ or $C = B^z$,

In Appendix B we discuss the trapdoor DDH group proposed by Dent and Galbraith [13], and also make preliminary observations about our assumptions above.

4 The New Construction

Our SHS construction resembles the construction of accumulators in [4, 16, 18, 7].

Representation of Elements. We shall require that the elements in the universe (a finite subset of which will be committed to) be represented by prime numbers. If the elements are originally represented otherwise, we need an efficient (and efficiently invertible) mapping from this universe to prime numbers. We remark that it is important for us that the representation used be unambiguous:

for any element there should be exactly one prime number representing it, and the prover and verifier should *a priori* agree on this.

Also we require that the primes used to represent the elements in the universe be "large." This depends on the security parameter and the trapdoor DDH group scheme we use. More precisely, with a security parameter k, it suffices to require these prime numbers to be all greater than $N = 2^k n$, where n is the bound on the order of $g \in \mathbb{G}$ generated by the trapdoor DDH group generating algorithm Gen.

In Appendix A we show that such a mapping can be easily achieved (but based on standard number theoretic conjectures). In the rest of the paper we shall identify the elements of the universe with the prime numbers representing themselves.

Construction 1 (Statistically Hiding Sets) The construction consists of the following four algorithms:

1. setup The verifier takes the security parameter k as the input and runs a trapdoor DDH group generation algorithm to obtain $(\mathbb{G}, g, n, \tau)$. The verifier then sends $(\mathbb{G}, g, n)$ to the prover. The prover verifies that $\mathbb{G}$ is indeed a group and $g \in \mathbb{G}$ has order at most n. (Recall that the order of g is explicitly included as part of description of $\mathbb{G}$. See Footnote 3.) In the following let $N = 2^k n$.

2. commit To commit to a set $S = \{p_1, p_2, \ldots, p_m\}$, the prover chooses an integer $v \in \{0, \ldots, N-1\}$ at random and computes

$$u = \prod_{i=1}^{i=m} p_i \qquad\qquad C = g^{uv}$$

   and outputs $\sigma = C$ as the public commitment and $\rho = uv$ as the private information for generating proofs in future.

3. prove When the prover receives a query about an element p, it will respond depending on whether $p \in S$ or not:

   - $p \in S$: In this case the prover computes $u_0 = u/p$ and $C_0 = g^{vu_0}$ and sends $\pi = (\mathsf{yes}; C_0)$ to the verifier.
   - $p \notin S$: Since $v < p$ and $\gcd(p, u) = 1$, we have $\gcd(p, uv) = 1$. The extended Euclidean algorithm allows the prover to compute integers (a_0, b_0) such that $a_0 uv + b_0 p = 1$. The prover then chooses a number $\gamma \in \{0, \ldots, N-1\}$ at random, and forms the pair of integers $(a, b) := (a_0 + \gamma p, b_0 - \gamma uv)$. Finally, the prover evaluates $A = g^a$, $B = g^b$, $D = C^a$ and sends $\pi = (\mathsf{no}; A, B, D)$ to the verifier.

4. verify When the verifier queries about an element p and receives a proof π from the prover, it does the following:

   - If π is of the form (yes, C_0), i.e., the prover asserts that $p \in S$, then the verifier checks if $C_0^p = C$ holds. It outputs 1 if the preceding equation holds, and $\perp$ otherwise.

- If π is of the form (no, A, B, D), i.e., the prover claims $p \notin S$, then the verifier checks if $\mathsf{TDDH}(A, C, D; \mathbb{G}, g, \tau) = 1$ (i.e., whether (g, A, C, D) is a DDH tuple in $\mathbb{G}$) and if $D \cdot B^p = g$ holds in the group. It outputs 0 if both the checks are satisfied, and $\perp$ otherwise.

Theorem 1 *Construction 1 is a secure statistically hiding set scheme.*

In the rest of this section we prove this theorem by showing that our construction satisfies the completeness, soundness and statistical hiding properties.

Completeness. For the completeness property, we need to show that an honest prover can always convince the verifier about the membership of the queried element.

Lemma 2 *Construction 1 satisfies the completeness property.*

PROOF: For any finite set S and any element p, we show that an honest prover's proof of membership or non-membership will be accepted by an honest verifier. Let $S = \{p_1, \ldots, p_m\}$, let u, v be computed as in the commitment phase of our construction and p be an element queried by the verifier.

If $p \in S$, then $p \mid uv$ and so uv/p is an integer. So the prover can indeed compute $C_0 = g^{uv/p}$ and send it to the verifier. The verifier will accept this proof since $C_0^p = C$.

If $p \notin S$, then $p \nmid u$. Note that since we require $p \geq N$ and $0 \leq v < N$, we have $p > v$. So $p \nmid uv$ and p being prime, $\gcd(p, uv) = 1$. So the prover can run the extended Euclidean algorithm and find $(a_0, b_0) \in \mathbb{Z} \times \mathbb{Z}$ such that $a_0 uv + b_0 p = 1$. For any integer γ (and in particular for $\gamma \in \{0, \ldots, N-1\}$), for $(a, b) = (a_0 + \gamma p, b_0 - \gamma uv)$, it holds

$$auv + bp = (a_0 + \gamma p)uv + (b_0 - \gamma uv)p = a_0 uv + b_0 p = 1.$$

The prover evaluates $A = g^a$, $B = g^b$, $D = C^a = g^{uva}$ and sends (A, B, D) to the verifier. Clearly the tuple $(g, A, C, D) = (g, g^a, C, C^a)$ is a DDH tuple and hence $\mathsf{TDDH}(A, C, D; \mathbb{G}, g, \tau) = 1$. Also

$$D \cdot B^p = C^a \cdot B^p = g^{(uv)a} \cdot g^{pb} = g^{auv+bp} = g. \tag{1}$$

Hence in this case also the verifier accepts the proof. □

Soundness. The soundness property guarantees that the scheme does not allow a malicious prover to equivocate about the membership of any element p in the committed set.

Lemma 3 *Under the strong RSA and the KEA-DH assumptions, construction 1 has the soundness property.*

PROOF: To violate the soundness of our construction, a corrupt prover $\mathcal{A}$ must make a commitment C and subsequently for some prime p give out valid proofs for membership and non-membership of p. That is, on input $(\mathbb{G}, g, n)$, $\mathcal{A}$ outputs (p, C, C_1, A, B, D) satisfying

$$C_1^p = C \tag{2}$$

$$\exists a \text{ s.t. } A = g^a, D = C^a \tag{3}$$

$$D \cdot B^p = g \tag{4}$$

We will show that the probability of this is negligible.

First note that from KEA-DH there is an extractor $\mathcal{E}$ corresponding to $\mathcal{A}$, such that with almost the same probability as $\mathcal{A}$ satisfies the above conditions $\mathcal{E}$ outputs $(\mathbb{G}, g, p, C, C_1, A, B, D, z)$ such that in addition to the above conditions, either $A = g^z$ or $C = g^z$.

1. In case z is such that $C = g^z$, we consider two cases.
   - If $\gcd(p, z) = 1$: Then using extended Euclidean algorithm on input (p, z) one can efficiently compute integers (α, β) such that $\alpha z + bp = 1$. That is, $g^{az+bp} = g$. Using equation (2), $C_1^p = g^z$ and so this can be rewritten as $C_1^{ap} g^{bp} = g$. That is by setting $x = C_1^\alpha g^\beta$, we have $x^p = g$.
   - If $\gcd(p, z) \neq 1$, then $p \mid z$ as p is a prime. Then z/p is an integer. Let $x = A^{z/p} \cdot B$; note that by substituting $C = g^z$ into equation (3) we get $D = A^z$, and hence by equation (4), $x^p = g$.
2. In case $A = g^z$, then in the subgroup generated by g, we have $x^a = x^z$ for all x (that is, a and z are congruent modulo the order of g). From equations (2) (3) and (4), we have

   $$g = D \cdot B^p = C^a \cdot B^p = (C_1^p)^a \cdot B^p = (C_1^a \cdot B)^p.$$

   Setting $x = C_1^a \cdot B = C_1^z \cdot B$ we have $x^p = g$.

Thus in all cases from the output of $\mathcal{E}$, one can efficiently find (x, p) such that $x^p = g$ and $p > 1$. This is possible with the same probability (up to negligible difference) as $\mathcal{A}$ succeeding in breaking the soundness. But by the strong RSA assumption this probability must be negligible, Hence the probability that $\mathcal{A}$ violates soundness is negligible. □

Statistical Hiding. The hiding property is statistical and unconditional. We shall show that the commitment as well as the proofs given by an honest prover are distributed (almost) independent of the set that is committed to. The only guarantee that is required from the setup information $(\mathbb{G}, g, n)$ sent out by the verifier is that the order of $g \in \mathbb{G}$ is bounded by n (as will be verified by the prover).

First we prove that the commitment is almost uniformly distributed in the subgroup $\langle g \rangle$. We use the following observation.

Lemma 4 *Let $\mathcal{U}_m$ be the uniform distribution over $\mathbb{Z}_m$. Let $\mathcal{U}_{N|m}$ be the distribution over $\mathbb{Z}_m$ obtained by sampling an element uniformly from $\mathbb{Z}_N$ and reducing it modulo m. Then the statistical distance between $\mathcal{U}_m$ and $\mathcal{U}_{N|m}$ is at most m/N.*

PROOF: $\mathcal{U}_{N|m}$ can be written as a convex combination of $\mathcal{U}_m$ and of the distribution of v mod m when v is chosen uniformly at random from $\{tm, \ldots, N-1\}$, where $t = \lfloor N/m \rfloor$. The weights used in the convex combination are tm/N and $(N - tm)/N$ respectively. So the statistical difference is at most $(N - tm)/N < m/N$. □

Lemma 5 *For any finite $S \subseteq \mathcal{U}_k$, and any valid setup public-key $\mathsf{PK} = (\mathbb{G}, g, n)$, the commitment C produced by $\mathsf{commit}(\mathsf{PK}, S)$ is almost uniformly distributed over $\langle g \rangle$.*

PROOF: The proof is a simple consequence of Lemma 4.

Note that the commitment C is distributed as g_1^v where v is distributed as $\mathcal{U}_{N|m}$ where $m = \mathrm{ord}(g_1)$. Here $g_1 = g^u$ where $u = \prod_{p_i \in S} p_i$. Since $p_i > n > \mathrm{ord}(g)$ we have $\gcd(u, \mathrm{ord}(g)) = 1$, and therefore $m = \mathrm{ord}(g_1) = \mathrm{ord}(g)$ and $\langle g^u \rangle = \langle g \rangle$. Then applying Lemma 4 with $N = 2^k n$ and $m \leq n$, we get that the statistical distance between the distribution of C and the uniform distribution over $\langle g \rangle$ is at most 2^{-k} which is negligible in the security parameter k. □

That means, we can define the distribution $\mathsf{dummycommit}$ from the definition of statistical hiding (Section 2) as $\mathsf{dummycommit}(\mathbb{G}, g, n) = \{\sigma \leftarrow \mathcal{U}_{\langle g \rangle} : (\sigma, \sigma)\}$, where $\mathcal{U}_{\langle g \rangle}$ is the uniform distribution over the subgroup of $\mathbb{G}$ generated by g. This is for $(\mathbb{G}, g, n)$ being a valid public-key for the setup; otherwise — i.e., if $\mathbb{G}$ is not a valid group or $\mathrm{ord}(g) > n$ — then commit will abort. Next we specify $\mathsf{dummyprove}_0$ and $\mathsf{dummyprove}_1$.

Lemma 6 *For any valid public-key* $\mathsf{PK} = (\mathbb{G}, g, n)$*, any finite* $S \subseteq U_k$*, and any commitment (public and private information)* $(\sigma, \rho) = (C, uv)$ *that can be produced by* $\mathsf{commit}(S, \mathsf{PK})$*, for each prime* $p > \mathrm{ord}(g)$ *there are distributions* $\mathsf{dummyprove}_0(\mathbb{G}, g, n, C, p)$ *and* $\mathsf{dummyprove}_1(\mathbb{G}, g, n, C, p)$ *such that for* $p \in S$ *(respectively,* $p \notin S$*), the proof of membership of* p *(respectively of non-membership),* $\mathsf{prove}(p, \rho, \mathsf{PK})$ *is distributed almost as* $\mathsf{dummyprove}_1(\mathbb{G}, g, n, C, p)$ *(respectively as* $\mathsf{dummyprove}_0(\mathbb{G}, g, n, C, p)$*).*

PROOF: $\mathsf{dummyprove}_1(\mathbb{G}, g, n, C, p)$ is just concentrated on the single unique element (yes, C_0) where $C_0 \in \mathbb{G}$ is such that $C_0^p = C$ (i.e., $C_0 = C^{p^{-1} \bmod \mathrm{ord}(g)}$). Clearly, the proof generated by the honest prover is exactly the same (though the prover does not compute $p^{-1} \bmod \mathrm{ord}(g)$ directly).

$\mathsf{dummyprove}_0(\mathbb{G}, g, n, C, p)$ is defined as $(\mathsf{no}, A', B', D')$ where $A' = g^{a'}, B' = g^{b'}, D' = C^{a'}$, with (a', b') distributed as follows. Let $m = \mathrm{ord}(g)$. Let c be the unique number in $\{0, \ldots, m-1\}$ such that $C = g^c$. Then (a', b') is distributed uniformly over the set

$$Z_{c,p,m} = \{(a', b') | a', b' \in \mathbb{Z}_m \text{ and } a'c + b'p \equiv 1 \pmod{m}\}.$$

We argue that the proof of $p \notin S$ produced by an honest prover (conditioned on $\mathsf{PK} = (\mathbb{G}, g, n)$ and commitment C), no matter which set $S \subseteq U_k \backslash \{p\}$ is used, is almost identically distributed as $\mathsf{dummyprove}_0(\mathbb{G}, g, n, C, p)$. For this, it is enough to show that (a, b) computed by prove is such that $(a \bmod m, b \bmod m)$ is distributed (almost) identically as (a', b') above – i.e., uniformly over $Z_{c,p,m}$.

Firstly, by Lemma 4, we have that for γ as sampled by the prover (i.e., $\gamma \leftarrow \{0, \ldots, N-1\}$), $\gamma \bmod m$ is close to being uniform over $\mathbb{Z}_m$ (the statistical difference being negligible in the security parameter). So we can ignore this difference, and assume that in fact the prover does sample γ such that $\gamma \bmod m$ is uniform over $\mathbb{Z}_m$.

Secondly, note that $uv \equiv c \pmod{m}$, since $C = g^{uv} = g^c$. Hence, when the prover uses the extended Euclidean algorithm on the pair of integers (uv, p) to find $(a_0, b_0) \in \mathbb{Z}^2$ we have that

$$a_0 c + b_0 p \equiv 1 \pmod{m}. \tag{5}$$

We remark that the input to the extended Euclidean algorithm uv, and hence its output (a_0, b_0), do depend on u and hence on the set S being committed. In fact, even $(a_0 \bmod m, b_0 \bmod m)$ depends on uv and not just on c.

But recall that the prover sets (a, b) to be $(a_0 + \gamma b, b_0 - \gamma a)$ where γ is sampled such that $\gamma \bmod m$ is (almost) uniform over $\mathbb{Z}_m$. We make the following observations:

1. $((a_0 + \gamma p) \bmod m, (b_0 - \gamma c) \bmod m) \in Z_{c,p,m}$ for every integer γ, by equation (5).

2. Since $\gcd(p, m) = 1$, if $((a_0 + \gamma_1 p) \bmod m, (b_0 - \gamma_1 c) \bmod m) = ((a_0 + \gamma_2 p) \bmod m, (b_0 - \gamma_2 c) \bmod m)$ then $\gamma_1 \equiv \gamma_2 \pmod m$. So for each distinct value of $\gamma \bmod m$ there is a unique element in $Z_{c,p,m}$ that $((a_0 + \gamma p) \bmod m, (b_0 - \gamma c) \bmod m)$ evaluates to.

3. Finally, $|Z_{c,p,m}| = m$ because for each value of $a \in \mathbb{Z}_m$ there is a unique $b = (1 - ac) \cdot p^{-1} \in \mathbb{Z}_m$ such that $(a, b) \in Z_{c,p,m}$. So for every $(a', b') \in Z_{c,p,m}$ there is exactly one $\gamma \in \mathbb{Z}_m$ such that $(a', b') = ((a_0 + \gamma p) \bmod m, (b_0 - \gamma c) \bmod m)$.

So we conclude that since the prover samples γ (almost) uniformly from $\mathbb{Z}_m$, (a, b) it computes is such that $(a \bmod m, b \bmod m)$ is indeed (almost) uniform over $Z_{c,p,m}$. □

To conclude we derive the following from the above (see proof in Appendix C).

Lemma 7 *Construction 1 satisfies the statistical hiding property.*

5 Comparison with Merkle-Tree based Constructions

As mentioned before, the only technique for constructing zero-knowledge sets prior to ours was based on Merkle-trees and mercurial commitments. In this section we mention how our approach compares with the previous approach in terms of functionality and efficiency. See Appendix D for a more detailed comparison.

In terms of functionality, we can obtain statistical hiding without the prover having to accumulate state upon each query. Secondly, we do not require a trusted setup as required by a mercurial commitment scheme.

In terms of the communication complexity and private storage, our scheme is more efficient. In terms of computational complexity our scheme is comparable to the Merkle-tree based approaches, somewhat less efficient in generating the proofs, but possibly more efficient in verifying the proofs. The actual complexities depend on the specifics of the mercurial commitment used for the latter, as well as of the group operations and TDDH of the trapdoor DDH group. In Table 1 (in Appendix D) we provide a summary.

Statistically Hiding Databases. Merkle-tree based constructions of zero-knowledge sets naturally extend to zero-knowledge database with little or no overhead. Our construction does not extend in this manner. However in Appendix D.1 we point out a simple way to use zero-knowledge sets (or statistically hiding sets) in a black-box manner to implement zero-knowledge databases (or statistically hiding databases, respectively).

6 Conclusion

We have introduced an alternate definition of zero-knowledge sets, called statistically hiding sets. We have also presented an alternate approach to constructing this primitive, which is markedly different from the Merkle-tree based one. Our construction achieves statistical hiding without the prover having to accumulate state. This opens up the possibility of achieving other extensions which seem to present inherent difficulties in the Merkle-tree based constructions: one possibility is of achieving a notion of updateability with better privacy than obtained in [22]. We have

considered only stand-alone security here; a natural question is to achieve security in more complex environments, as is done for instance in [17].

The assumptions employed here are relatively new and need to be explored further. The knowledge of exponent assumption (KEA-DH) that we have introduced may be of independent interest.

References

[1] M. Abe and S. Fehr. Perfect NIZK with adaptive soundness. In *TCC*, pages 118–136, 2007.

[2] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO*, pages 255–270, 2000.

[3] G. Ateniese and B. de Medeiros. Efficient group signatures without trapdoors. In *ASIACRYPT*, pages 246–268, 2003.

[4] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology – EUROCRYPT '97*, pages 480–494, 1997.

[5] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In M. Franklin, editor, *Advances in Cryptology - Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[6] J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology – EUROCRYPT '93*, pages 274–285, 1994.

[7] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology - Crypto 2002*, volume 2442 of *LNCS*, pages 61–76. Springer Verlag, 2002.

[8] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *CRYPTO*, pages 413–430, 1999.

[9] D. Catalano, Y. Dodis, and I. Visconti. Mercurial commitments: Minimal assumptions and efficient constructions. In *Theory of Cryptography -TCC'06*, volume 3876 of *Lecture Notes in Computer Science*, pages 120–144. Springer-Verlag, 2006.

[10] M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. In *Advances in Cryptology – EUROCRYPT '05*, pages 422–439, 2005.

[11] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS)*, pages 46–51, Nov. 1999.

[12] I. Damgard. Towards practical public-key cryptosystems provably-secure against chosen-ciphertext at- tacks. In J. Feigenbaum, editor, *Advances in Cryptology–CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer-Verlag, 1992.

[13] A. W. Dent and S. D. Galbraith. Hidden pairings and trapdoor DDH groups. In S. P. F. Hess and M. Pohst, editors, *ANTS-VII*, volume 4076 of *LNCS*, pages 436–451. Springer, 2006.

[14] E. Fujisaki and T. Okamoto. Statistical zeroknowledge protocols to prove modular polynomial relations. In Jr.Burton and S.Kaliski, editors, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *LNCS*, pages 16–30. Springer-Verlag, 1997.

[15] S. D. Galbraith and J. F. McKee. Pairings on elliptic curves over finite commutative rings. In N. P. Smart, editor, *Cryptography and Coding: 10th IMA International Conference*, volume 3796 of *LNCS*, pages 392–409, Cirencester, UK, 2005. Springer.

[16] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT '99*, pages 123–139, 1999.

[17] R. Gennaro and S. Micali. Independent zero-knowledge sets. In *ICALP (2)*, pages 34–45, 2006.

[18] M. T. Goodrich, R. Tamassia, and J. Hasic. An efficient dynamic and distributed cryptographic accumulator. In *Proceedings of the 5th International Conference on Information Security*, pages 372–388, 2002.

[19] A. Granville. Harold cramér and the distribution of prime numbers. *Scandanavian Actuarial Journal*, 1:12–28, 1995.

[20] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In H. Krawczyk, editor, *Advances in Cryptology–CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423. Springer-Verlag, 1998.

[21] S. Hohenberger. The cryptographic impact of groups with infeasible inversion. Master's thesis, MIT, 2003.

[22] M. Liskov. Updatable zero-knowledge databases. In *Advances in Cryptology – ASIACRYPT '05*, pages 174–198, 2005.

[23] S. Micali, M. Rabin, and J. Kilian. Zero-knowledge sets. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, page 80, 2003.

[24] D. Mireles. An attack on disguised elliptic curves. Cryptology ePrint Archive, Report 2006/469, 2006. `http://eprint.iacr.org/`.

[25] R. Ostrovsky, C. Rackoff, and A. Smith. Efficient consistency proofs for generalized queries on a committed database. In *ICALP*, pages 1041–1053, 2004.

[26] R. L. Rivest. On the notion of pseudo-free groups. In M. Naor, editor, *TCC 2004 (Copyright IACR.)*, volume 2951 of *Lecture Notes in Computer Science*, pages 505–521. Springer-Verlag, 2004.

A Mapping Strings to Primes

Our construction requires that the elements in the universe (a finite subset of which will be committed to) be represented by prime numbers. If the elements are originally represented otherwise, we need a mapping from this universe to prime numbers, with the following properties:

- Unambiguity: no two strings map to the same prime number, and each string has exactly one prime number that it maps to.
- Efficiency: the prime number corresponding to a string can be computed in time polynomial in the length of the string.
- Efficient invertability: given a (prime) number, it can be determined whether any string maps to it, and if so which one, all in time polynomial in the representation length of the number.

Several approaches for representing bit strings as primes are suggested in the accumulator literature [4, 16]. However, these mappings are not unambiguous (but it suffices for the purposes in these works, as accumulators are concerned only with proving membership, and not non-membership). So instead we point out that based on standard conjectures on the density of prime numbers, simpler mappings have the desired properties.

We shall consider the problem of mapping binary strings of length polynomial in k into prime numbers,[4] where k is the security parameter (and all efficient operations should be in time polynomial in k, with failure probability negligible in k). Consider the following mapping: first map the given binary string unambiguously to a sufficiently large integer t (but not larger than $2^{\text{poly}(k)}$).[5] Then map it to *the smallest prime number greater than* t^2.

To argue that this mapping has the desired properties, we rely on a widely-believed conjecture on the density of prime numbers, which is supported by empirical evidence.

Conjecture 1 (Cramer-Granville Conjecture [19]) *$p_{n+1} - p_n = O(\log^2 p_n)$, where p_n stands for the n^{th} largest prime number.*

(The exact power of the logarithm in this conjecture is not important for us.)

By this conjecture, for sufficiently large t, there is a prime in the range $[t^2, (t+1)^2)$. This guarantees unambiguity. Further, it also guarantees that finding the smallest prime number greater than t^2 can be done in time polynomial in $\log t$ (using a polynomial time primality test to test each integer starting from t^2, until a prime is found). Finally, the mapping is efficiently invertible, since given a (prime) number p, we can easily find the largest square $t^2 \leq p$ (say by binary search), and then find the smallest prime $p' \geq t^2$ as before. If $p \neq p'$, then p is not in the range of the map, and otherwise t maps to p.

B On the Assumptions

Trapdoor DDH groups. Trapdoor DDH groups were recently introduced by Dent and Galbraith [13]. To the best of our knowledge it has not been employed in any cryptographic application (except an simple illustrative example in [13]). Dent and Galbraith also gave a candidate for this primitive based on an elliptic curve group over the ring $\mathbb{Z}/\mathbb{Z}_N$ with $N = q_1 q_2$ being the product of two large random primes. The public information $\mathbb{G}$ produced by the TDG generation algorithm

[4] For our application, to reflect the parameters listed in Table 1, we will require that these binary strings are in fact of length *linear* in k, so that the prime numbers obtained also have $O(k)$ number of bits. If universe U_k has longer strings, we can first use a collision resistant hash function to hash the elements down to a k bit string.

[5] Recall that we have a lowerbound on the primes that can be mapped to. t will be chosen to be large enough to ensure that our mapping yields only such primes. Further t is chosen to be larger than some constant, so that the asymptotic behavior required for correctness sets in.

Gen consists of N and a fixed equation describing the elliptic curve.a The element g produced by Gen is an element in $\mathbb{G}$ of order $m = r_1 r_2$, where $r_1 \mid q_1 + 1$, $r_2 \mid q_2 + 1$ and r_1, r_2 are large primes. The trapdoor produced by Gen is $\tau = (q_1, q_2, r_1, r_2)$. The trapdoor DDH algorithm TDDH uses the trapdoor information to carry out a bilinear pairings, like Weil or Tate pairing, on the elliptic curve groups over $\mathbb{F}_{q_1}$ and $\mathbb{F}_{q_2}$ separately, and (by Chinese Remainder Theorem) concludes if a given tuple is a DDH tuple in the group $\mathbb{G}$ (using two pairings each in the two groups).

Stong RSA. The Strong RSA assumption was introduced by Barić and Pfitzmann in their work on accumulators [4] mentioned above. Subsequently it has been studied extensively and used in a variety of cryptographic applications [16, 11, 8, 2, 7, 3, 26]. The original strong RSA assumption was proposed for a specific family of groups, namely $\mathbb{Z}_N$ where N is generated as the product of two large random primes. Rivest [26] showed that the strong RSA assumptions easily holds in a *pseudo-free group* [21], an abstraction capturing the intuition behind many typical hardness assumptions from algorithmic number theory. Rivest conjectures that the family $\mathbb{Z}_N$ (generated as above) is in fact pseudo-free. We propose a natural extension of this conjecture, that the TDG family from [13] described above is pseudo-free. This conjecture is stronger than the strong RSA assumption that we need.

Knowledge-of-Exponent Assumption. The second assumption we make is a Knowledge-of-Exponent Assumption called KEA-DH. This is similar in spirit to – but stronger than – the assumption KEA1 introduced by [12]. Informally, KEA1 asserts that when given a group $\mathbb{G}$ and an element g (generated according to a specified distribution) and g^a where a is picked at random (from $\mathbb{Z}_{\mathrm{ord}(g)}$), to create a DDH tuple (g, g^a, g^b, g^{ab}) an adversary should necessarily know b. KEA-DH allows the adversary to choose g^a as well, but asserts that if the adversary does not know a, then KEA1 still holds as if a was picked at random: that is, the adversary should know either a or b to be able to produce the DDH tuple (g, g^a, g^b, g^{ab}).

In justifying the KEA-DH assumption it is important that the trapdoor DDH group $\langle g \rangle \subseteq \mathbb{G}$ is not the *target group* of a bilinear pairing. Indeed, suppose there is a bilinear pairing $e : \mathbb{G}' \times \mathbb{G}' \to \mathbb{G}$ with $e(g'^a, g'^b) = g^{ab}$. Then by taking two elements $A', B' \in \mathbb{G}'$ and setting $A = e(g', A')$, $B = e(g', B')$, $C = e(A', B')$, an adversary can form a DDH tuple (g, A, B, C). (However, for such an attack to be possible, the adversary should be able to efficiently find – and evaluate – the bilinear pairing e, for a randomly generated g.) The construction of Dent and Galbraith has $\mathbb{G}$ as the source group (and not the target group) of a bilinear pairing. Further, even if there is a bilinear pairing as above, since the factorization of the order of $\mathbb{G}$ (or of $\langle g \rangle$) is not known to the adversary, it is unlikely that an adversary can exploit the existence of a bilinear pairing operation. As such, this kind of an attack is not an immediate concern for us.

Assuming hardness of the Discrete Logarithm problem (in a strong sense; see footnote 6), it is easy to see that KEA-DH is indeed stronger than KEA1 An extractor for an adversary $\mathcal{A}_1$ as required by KEA1 can be constructed from the extractor $\mathcal{E}_{\mathsf{DH}}$ for an adversary $\mathcal{A}_{\mathsf{DH}}$ as guaranteed by KEA-DH. Here $\mathcal{A}_{\mathsf{DH}}$ works as follows: on receiving input $(\mathbb{G}, g)$, $\mathcal{A}_{\mathsf{DH}}$ samples a random element[6] $A \in \mathbb{G}$ and sends $(\mathbb{G}, g, A)$ as input to $\mathcal{A}_1$. Then $\mathcal{A}_{\mathsf{DH}}$ outputs the tuple (g, A, B, C) returned by

[6]Here we assume that such directing sampling is allowed by the representation of the group $\mathbb{G}$. We further assume that any element in the group can be explained (using a random tape) as the outcome of such a sampling. We will also require that the Discrete Logarithm problem remains hard even when the randomness used to sample the group element is given.

$\mathcal{A}_1$. Now if it is indeed a DDH tuple, then the extractor $\mathcal{E}_{\mathsf{DH}}$ (with the randomness of $\mathcal{A}_{\mathsf{DH}}$ and $\mathcal{A}_1$ as inputs), outputs α such that $g^\alpha = A$ or $g^\alpha = B$. However, by the hardness of the Discrete Logarithm problem, probability for the first event is negligible, and hence the output α is such that $g^\alpha = B$. Hence an extractor $\mathcal{E}_1$ which constructs a random-tape for $\mathcal{A}_{\mathsf{DH}}$ (explaining the input A as sampled by an $\mathcal{A}_{\mathsf{DH}}$), and invokes $\mathcal{E}_{\mathsf{DH}}$ on this functions as KEA1 extractor for $\mathcal{A}_1$.

B.1 KEA-DH Holds in the Generic Group Model

We illustrate the plausibility of the KEA-DH assumption by proving that it holds in trapdoor DDH a group as long as only "generic group operations" are used (and the order of the group has at least one large prime factor). The generic group, for us, is endowed with a bilinear pairing operation, and the adversary has access to a DDH oracle. The proof is a simple extension to similar arguments elsewhere; in particular see [1].

In the generic group model, an adversary can interact with a blackbox implementing the operations associated with the groups. In all communications, group elements are represented using handles (usually modeled as random strings, or can be modeled as arbitrary strings provided by the adversary). The blackbox internally maintains a table associating the handles it gives out with the group elements. (It will ensure that the same group element is not assigned two different handles, and same handle is not used for two group elements.)

The blackbox first provides a handle for g, a generator of the group. It then provides the following operations: randomly sample an element in the group and return a handle for it, multiply two group elements (taking two handles as input, and returning a handle to their product), or raise a group element to an integer (taking a handle and an integer as input, and outputting a handle). Later we consider when a bilinear pairing operation (taking two handles and returning a handle for an element in the target group) is also provided by the blackbox.

A KEA-DH assumption essentially states that if an adversary $\mathcal{A}$ takes a handle for an element g and returns three handles (A, B, C) that form a DDH tuple with respect to the base g, then an extractor can return an exponent z such that either $A^z = C$ or $B^z = C$. We will show that the following extractor $\mathcal{E}_{\mathcal{A}}$ works: it monitors the interaction between $\mathcal{A}$ and the blackbox; if A or B was obtained from the blackbox as g^z (z being an integer), then the extractor returns z, and otherwise it fails. If the extractor does not fail, then clearly, (A, B, C) for a DDH tuple only if $C = A^z$ or $C = B^z$. Hence we are interested in bounding the probability that the extractor fails, and (A, B, C) is a DDH tuple. Let us denote this event by $\mathsf{fail}_{\mathcal{E}_{\mathcal{A}}}$.

For each handle, consider its discrete logarithm (to the base g). Representing the discrete logarithm of handles obtained by the sampling operation by formal variables x_i (i being the i-th sampling). Then the discrete logarithms of any handle can be associated with a linear multi-variate polynomial in these variables. We will denote the polynomial for a handle X by p_X. (Then if the handle returned for the i-th sampling is X_i, we have p_{X_i} be simply the polynomial x_i. In general p_X is a polynomial in $x_1, \ldots, x_m$ where m is an upperbound on the number of accesses to the blackbox.) For any adversary $\mathcal{A}$ we can define an adversary $\mathcal{A}'$ which runs $\mathcal{A}$ and monitors its interaction with the blackbox, and outputs the polynomial $p_X - p_{X'}$ for every two handles X, X' that $\mathcal{A}$ receives from the blackbox. Further when $\mathcal{A}$ outputs a purported DDH tuple (A, B, C), $\mathcal{A}'$ outputs the polynomial $p_A p_B - p_C$.

To bound the maximum probability of $\mathsf{fail}_{\mathcal{E}_{\mathcal{A}}}$ (maximized over all adversaries which make at most m accesses of the blackbox), we define the following event called $\mathsf{succ}_{\mathcal{A}'}$. $\mathsf{succ}_{\mathcal{A}'}$ is said to occur if an adversary $\mathcal{A}'$ interacting with the blackbox outputs a non-zero polynomial which evaluates to

zero modulo the order of the group q, when the variables x_i are assigned values at random from $\mathbb{Z}_q$.[7]

Note that if the tuple (A, B, C) that the adversary $\mathcal{A}$ outputs indeed forms a DDH tuple, then the polynomial $p_A p_B - p_C$ that $\mathcal{A}'$ outputs evaluates to zero for the particular choice of values for X_i that the sampling makes. Further if this polynomial is uniformly zero, then, since p_A, p_B, p_C are all linear, at least one of p_A and p_B should be a constant z, and the extractor $\mathcal{E}_{\mathcal{A}}$ does not fail. So the maximum – over all adversaries that make at most m blackbox accesses – of the probability of $\mathsf{fail}_{\mathcal{E}_{\mathcal{A}}}$ is upperbounded by the maximum (over the same class of adversaries) of the probability of $\mathsf{succ}_{\mathcal{A}'}$.

Now we bound the probability that $\mathsf{succ}_{\mathcal{A}'}$ occurs. For the sake of rigour, first we consider modifying the blackbox so that it does not assign values to the random samples, but maintains only formal variables; it gives the same handle to two polynomials only if they are identical polynomials. At the end it picks random values for all the variables. The probability that the event $\mathsf{succ}_{\mathcal{A}'}$ occurs is not altered when the blackbox is modified in this way: to see this consider a run of the interaction of $\mathcal{A}'$ with both the blackboxes responding to it (with a shared state, and the same assignment of variables). The two blackboxes diverge only at the point where the adversary succeeds in producing a non-zero polynomial which evaluates to zero for the chosen assignment of variables. However, when this occurs the event $\mathsf{succ}_{\mathcal{A}'}$ occurs in both the experiments.

Now, with the modified blackbox, the probability of $\mathsf{succ}_{\mathcal{A}'}$ is easy to bound. Firstly, the probability that say the t-th polynomial (of degree 1 or 2) output by $\mathcal{A}'$ evaluates to zero for the randomly chosen assignment is at most $O(1/q')$, where q' is the largest prime factor of q. (This follows by the standard Schwatrz-Zippel lemma and the Chinese remainder theorem.) Since $\mathcal{A}'$ outputs at most m^2 such polynomials (because $\mathcal{A}$ gets at most m handles), the probability that any of them evaluates to zero is $O(m^2/q')$ by the union bound. Hence we conclude that the probability of $\mathsf{succ}_{\mathcal{A}'}$ and hence of $\mathsf{fail}_{\mathcal{E}_{\mathcal{A}}}$ is $O(m^2/q')$ which is negligible if q' is exponentially large in the security parameter. For instance, if q is the product of two primes, we have $q' > \sqrt{q}$, and hence our bound is $O(m^2/\sqrt{q})$.

Now we consider extending this analysis to model a generic bilinear pairing operation as well. This corresponds to considering polynomials $\bar{p}_Y$ as well, where Y is a handle for an element in the *target group* of the bilinear operation. In this case the polynomial $\bar{p}_Y$ can be quadratic (obtained as the product of two polynomials p_{X_1} and p_{X_2}). Even with this addition, the above arguments go through verbatim, by redefining $\mathcal{A}'$ to output polynomials $\bar{p}_Y - \bar{p}_{Y'}$ for handles Y, Y' for the target group elements (as well as $p_X - p_{X'}$ for handles X, X' for the source group elements), and redefining the event $\mathsf{succ}_{\mathcal{A}'}$ accordingly.

Finally we enhance the blackbox with a DDH oracle. In the above analysis, we modify this blackbox with one which implements the DDH oracle using a test on polynomials (i.e., on input a triple of handles (X, Y, Z) it checks if $p_X p_Y = p_Z$). This modification does not alter the probability of $\mathsf{succ}_{\mathcal{A}'}$ (because, as before, until the event $\mathsf{succ}_{\mathcal{A}'}$ occurs, the two blackboxes behave identically). Then the probability of $\mathsf{succ}_{\mathcal{A}'}$ is the same as before, namely $O(m^2/q')$.

[7] The adversary may or may not know q. When considering a real group as in our construction, it could be crucial that the adversary does not know q. However, once we are in the generic group model, it is not important anymore. We point out that the extractor we defined does not use q.

C Proof of Lemma 7

PROOF: This follows from Lemmas 5 and 6. The two lemmas give distributions $\mathsf{dummycommit}(\mathsf{PK})$, and $\mathsf{dummyprove}_0(\mathsf{PK}, \rho, x)$ and $\mathsf{dummyprove}_1(\mathsf{PK}, \rho, x)$, as required in the definition of statistical hiding property. That they indeed statisfy the condition there, is an easy consequence of the fact that the adversary $\mathcal{A}$ is allowed auxiliary information about the set S in the experiments.

More formally, we define $t + 1$ hybrid distributions, by modifying the experiments in the two distributions in the definition of the hiding property: commit is used to generate the commitment, but in the i^{th} hybrid, for the first $i - 1$ iterations of the `for` loop dummyprove is used,[8] and in the rest prove is used. The first of these hybrids is identical to the left hand side distribution in the definition, and the last one is statistically close to the right hand side distribution by Lemma 5. Two adjacent hybrids are statistically close by Lemma 6. In applying Lemma 5 we use an adversary which has S itself as auxiliary information, and in applying Lemma 6 we can use adversaries which have S as well as the (public and private) commitment information ($\sigma = C$ and $\rho = uv$) as auxiliary information, so that these adversaries can carry out the experiments themselves. Note that the lemmas do allow this conditioning. □

D Comparison with Merkle-Tree based Constructions

As mentioned before, the only technique for constructing zero-knowledge sets prior to ours was based on Merkle-trees and mercurial commitments. In this section we compare our approach to this previous approach in terms of functionality and efficiency.

In terms of functionality, there are a few advantages that our construction enjoys over the Merkle-tree based constructions. Firstly, we can obtain statistical hiding without the prover having to accumulate state upon each query. Secondly, we do not require a trusted setup as required by a mercurial commitment scheme. The setup algorithm in our case can be carried out by the verifier (the hiding property does not depend on the correctness of this step).

In terms of the communication complexity and private storage, our scheme is more efficient. In terms of computational complexity our scheme is comparable to the Merkle-tree based approaches, somewhat less efficient in generating the proofs, but possibly more efficient in verifying the proofs. The actual complexities depend on the specifics of the mercurial commitment used for the latter, as well as of the group operations and TDDH of the trapdoor DDH group. In Table 1 we provide a summary. It is not meant to be a rigorous comparison, as we do not explore the complexity of the various operations involved (mercurial commitments, group operations etc.). Also note that the security parameters may not be directly comparable in the two cases. Below we explain the table further.

The Merkle-tree based zero-knowledge sets employ a collision resistant hash function, mercurial commitments and pseudorandom functions. Our approach employs a trapdoor DDH group, and the prover accesses the group operation, while the verifier accesses both the group operation and the trapdoor DDH operation associated with it. The comparison in Table 1 is in terms of the number of these operations. k stands for the security parameter, and the size of representation of all elements (inputs/outputs to the hash function, the mercurial commitments and pseudorandom function, as well as the group elements in our construction) are assumed to be linear in k. Also, m

[8] Note that dummyprove takes C as an input; C produced by commit will be given.

		Merkle-tree based scheme	TDG based scheme
Size (in bits)	$\|\sigma\|$	$O(k)$	$O(k)$
	$\|\rho\|$	$O(mk^2)$	$O(k)$
	$\|\pi\|$	$O(k^2)$	$O(k)$
Operations	commit	$O(mk)$ hashes + $O(mk)$ com	$O(mk)$ g.o.
	prove.yes	$O(1)$ (table lookup)	$O(mk)$ g.o.
	prove.no	$O(k)$ hashes + $O(k)$ decom + $O(k)$ PRF	$O(mk)$ g.o. + $O(mk^2)$
	verify.yes	$O(k)$ hashes + $O(k)$ decom-ver	1 exp
	verify.no	$O(k)$ hashes + $O(k)$ decom-ver	1 TDDH + 1 exp

Table 1: *Comparing the size complexity and number of operations. σ and ρ are the public and private information produced by* commit*. π is the proof produced by* prove*. Com, decom and decom-ver refer to operations that are part of a mercurial commitment scheme; hashes refer to applications of the CRHF, and PRF to the invoking the pseudorandom function; g.o. stands for the group operation and* TDDH *for the trapdoor DDH algorithm as part of the* TDG*; exp stands for an exponentiation carried out with the knowledge of the trapdoor for the* TDG*, and is not more expensive than $O(k)$ group operations.*

stands for the size of the set being committed. For our TDG based implementation note that we list most of the computational complexity in terms of the number of group operations. This takes into account the exponentiations carried out (using repeated squaring). For operations carried out by the verifier we list the operations in terms of exponentiation (rather than $O(k)$ group operations), as the verifier may be able to use its trapdoor to carry out the exponentiation more efficiently. The sizes of the commitment σ, the private information stored ρ and the sizes of the proofs π are all given in number of bits.

Where our scheme is less efficient than the Merkle-tree based constructions is in the computation involved in generating proofs. This is because the prover does not know the order of the base g to which the exponentiations are carried out. As a result the prover cannot reduce the exponents modulo the order, nor can it find p^{-1} for any exponent p. As a consequence, firstly, we cannot gain much by precomputing the proofs of membership while computing the commitment. Secondly, since the exponents are large numbers (in the order of $2^{\Theta(mk)}$), running the extended Euclidean algorithm, or raising g to these exponents, incur relatively large computational costs, which depend on m. In constrast, the complexity of prove is independent of m in the case of Merkle-tree based constructions.

D.1 Zero-Knowledge/Statistically Hiding Databases

A zero-knowledge database is an extension of a zero-knowledge set, in which each element (*key*) present in the database has a *value* associated with it. When the verifier queries a key the prover can answer with either $\perp$ (indicating that the key is not in the database) or, if the key is in the database, then the value that is associated with it. Merkle-tree based constructions of zero-knowledge sets naturally extend to zero-knowledge database with little or no overhead. Our construction does not extend in this manner.

However we point out a simple way to use zero-knowledge sets (or statistically hiding sets) in a black-box manner to implement a zero-knowledge database (respectively statistically hiding sets) but with some overhead. To commit to a database D of pairs of the form (x, y) (x being the key and y the value), the prover constructs two sets: $S_{key} = \{x|(x, y) \in D\}$ and $S_{data} = \{x \circ y|(x, y) \in D\}$, where $x \circ y$ stands for an unambiguous encoding of (x, y). Then the prover invokes the zero-knowledge set (or SHS) scheme twice to commit to S_{key} and S_{data} separately. On query x, if x is a key in D, then the prover responds with y such that $(x, y) \in D$; it also invokes the membership proofs in the two instances of zero-knowledge set (or SHS) scheme to prove $x \in S_{key}$ and $(x, y) \in S_{data}$. If x is not a key in the database, the prover gives a proof that $x \notin S_{key}$.

Note that in this scheme, the database does not require the value associated with a key to be unique. However, if the value comes from a small universe, databases which require a unique value for a key can also be supported: for this a proof that $x \in S_{key}$ and $(x, y) \in S_{data}$ will be augmented by a proof that $(x, y') \notin S_{data}$ for every possible value $y' \neq y$.