Sufficient Conditions for Collision-Resistant Hashing

Yuval Ishai[1⋆], Eyal Kushilevitz[1⋆⋆], and Rafail Ostrovsky[2⋆⋆⋆]

[1] Computer Science Department, Technion, Haifa 32000, Israel.
{yuvali,eyalk}@cs.technion.ac.il
[2] Computer Science Department, UCLA.
rafail@cs.ucla.edu

Abstract. We present several new constructions of *collision-resistant hash-functions* (CRHFs) from general assumptions. We start with a simple construction of CRHF from any *homomorphic encryption*. Then, we strengthen this result by presenting constructions of CRHF from two other primitives that are implied by homomorphic-encryption: one-round *private information retrieval* (PIR) protocols and *homomorphic one-way commitments*.

Keywords. Collision-resistant hash functions, homomorphic encryption, private information-retrieval.

1 Introduction

Collision resistant hash-functions (CRHFs) are an important cryptographic primitive. Their applications range from classic ones such as the "hash-and-sign" paradigm for signatures, via efficient (zero-knowledge) arguments [14, 17, 2], to more recent applications such as ones relying on the non-black-box techniques of [1].

In light of the importance of the CRHF primitive, it is natural to study its relations with other primitives and try to construct it from the most general assumptions possible. It is known that CRHFs can be constructed from claw-free pairs of permutations [5] (which in turn can be based on the intractability of discrete logarithms or factoring) and under lattice-based assumptions [10]. On the other hand, Simon [20] rules out a black-box construction of CRHF from one-way permutations; thus, there is not much hope to base CRHF on very general assumptions involving one-wayness alone.

In practice, when people are in need for CRHFs in various cryptographic protocols, they often use constructions such as SHA1, MD5 and others. However,

⋆ Partially supported by Israel Science Foundation grant 36/03.
⋆⋆ Partially supported by BSF grant 2002-354 and by Israel Science Foundation grant 36/03.
⋆⋆⋆ Partially supported by BSF grant 2002-354 and by a gift from Teradata, Intel equipment grant, OKAWA research award and NSF Cybertrust grant.

recent weaknesses found in some of these constructions (such as MD5) [22] only provide further evidence for the value of theoretically sound constructions.

Our results. In this paper, we present several new constructions of CRHFs from general assumptions. We start by describing a simple construction of CRHF from any *homomorphic encryption* (HE) scheme. A homomorphic encryption is a semantically secure encryption in which the plaintexts are taken from a group, and given encryptions of two group elements it is possible to efficiently compute an encryption of their sum. For instance, the Goldwasser-Micali scheme [11] is homomorphic over the group Z_2. We note that this notion does not impose any algebraic structure on the space of ciphertexts, but only on the space of plaintexts.

We then weaken the above assumption in two incomparable ways. First, we show how to construct CRHF from any (single-server, sublinear-communication) one-round PIR protocol [15]. Since PIR is implied by homomorphic encryption [15, 21, 16], this is indeed a weaker assumption. This result strengthens the result of [3], that constructs unconditionally hiding commitment (UHC) from PIR, as it is known that CRHF imply UHC [6, 12].

Second, we obtain a construction of CRHFs from *homomorphic one-way commitments* (HOWC). Such a commitment does not provide semantic security for the committed value x but only "one-way" security, guaranteeing that x is hard to find. For instance, a simple deterministic HOWC is defined by $C(x) = g^x$, where g is a generator of a group in which finding discrete logarithms is hard.

The relation between the different primitives discussed above is summarized in Figure 1.

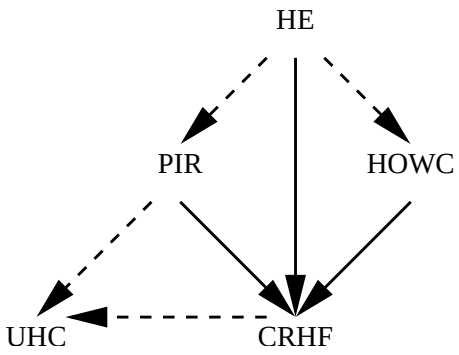

Fig. 1. Solid arrows stand for implications shown in this paper. Dashed arrows stand for implications that were shown in other papers (or that follow directly from the definition).

One way to think of our results is the following. It is known how to build CRHFs from all major (specific) assumptions used in public-key cryptography.

These (specific) assumptions also have an algebraic structure that usually implies homomorphic properties. The results of this work suggest that this is not a coincidence, establishing a rather general link between "homomorphic" properties and collision resistance. First results in this direction were given in [18]; see below.

Related Work. As mentioned, Damgård [5] shows how to construct CRHFs based on any claw-free pair of permutations (and based on specific assumptions such as the hardness of factoring or discrete-log). Russell [19] shows that this is essentially the best one can do, as the existence of CRHFs is equivalent to the existence of a related primitive that he terms "claw-free pair of pseudo-permutations"; this characterization is not satisfactory in the sense that this primitive is not a well-studied one and its relations with other primitives are not known. Hsiao and Reyzin [13] consider two variants of the definition of CRHF (that differ in whether or not the security of the CRHF depends on the secrecy of the random coins used by the key-generation algorithm) and show some relations between the two variants. Simon [20] shows, by demonstrating an appropriate separation oracle, that one-way permutations are unlikely to imply CRHFs (see [7] for some stronger versions of this result). In contrast with [20], Ogata and Kurosawa [18] show that a stronger version of one-way permutations, i.e. *homomorphic one-way permutations*, can be used to construct claw-free permutations and hence also CRHFs. While this result gives an indication for the usefulness of homomorphic properties for constructing CRHFs, their construction heavily relies on the function being a *permutation*; our results, on the other hand, do not impose such structural constraints on the underlying primitives. To illustrate the significance of the extra generality, consider the question of basing CRHF on lattice-related intractability assumptions. Combining our results with the lattice-based PIR scheme from [16], we can obtain CRHFs whose security is based on a standard lattice-related assumption (providing an alternative to [10]). In contrast, there are no known constructions of one-way *permutations* (let alone homomorphic ones) from such assumptions.

Organization. In Section 2, we provide some necessary definitions (in particular that of CRHF). The first construction of CRHF, presented in Section 3, is based on the existence of homomorphic encryption. In Sections 4 and 5, we strengthen this result by describing constructions that are based on (computational) PIR and on homomorphic one-way commitment (respectively).

2 Preliminaries

We start with a formal definition of collision-resistant hash-functions (CRHFs). In fact, the definition applies to a *family* of functions,[1] and uses the terminology of secret-coin CRHFs from [13].

[1] Speaking of a *single* collision-resistant is meaningless if one allows the adversary to be non-uniform.

Definition 1. *Let $\ell, \ell' : \mathbb{N} \to \mathbb{N}$ be such that $\ell(n) > \ell'(n)$ and let $I \subseteq \{0,1\}^*$. A collection of functions $\{H_s\}_{s \in I}$ is called* (secret-coin) collision-resistant hash family *(with index-set I) if the following holds:*

1. *There exists a probabilistic polynomial-time* key-generation *algorithm,* Gen*, that on input 1^k outputs an* index *$s \in I$ (of a function H_s). The function H_s maps strings of length $\ell(k)$ to strings of length $\ell'(k)$.*
2. *There exists a probabilistic polynomial-time* evaluation *algorithm that on input $s \in I, x \in \{0,1\}^{\ell(k)}$ computes $H_s(x)$.*
3. *Collisions are hard to find. Formally, a pair x, x' is called a* collision *for a function H_s if $x \neq x'$ but $H_s(x) = H_s(x')$. The collision-resistance requirement states that every probabilistic polynomial-time algorithm B, that is given input $s = \text{Gen}(1^k)$, succeeds in finding a collision for the function H_s with a negligible probability (where the probability is taken over the coin tosses of both* Gen *and B).*

Remark 1. Various variants of the above definition are possible. For example, one can consider hash-functions that can be applied to strings of arbitrary length (and not just to strings of the specified length $\ell(k)$); such functions can be obtained from the more restricted functions, defined above, by using standard techniques such as block-chaining or hash-trees (where the restricted function is applied repeatedly); cf. [9, Sec. 6.2.3].

Example 1. Let p be a prime and q be a "large" divisor of $p-1$. Let $h_1, h_2 \in Z_p^*$ be two elements of order q. Let $H_{p,h_1,h_2}(x) = h_1^{x_L} \cdot h_2^{x_R} \bmod p$, where $x = (x_L, x_R) \in Z_q \times Z_q$, and consider the family of all these functions. Each such function maps strings of length $2 \log q$ to strings of length $\log p$. An algorithm that can find a collision for such a function (i.e., x, x' such that $H_{p,h_1,h_2}(x) = H_{p,h_1,h_2}(x')$) can be used to compute the discrete-log $\text{DLOG}_{h_1}(h_2) \bmod p$.

3 CRHF from Homomorphic Encryption

In this section, we present the simplest of our constructions. This construction is based on a stronger assumption than what we use in subsequent sections; namely, the existence of *homomorphic encryption* schemes. In fact, we never use the standard requirement from encryption schemes that decryption can also be performed in polynomial-time (but just the fact that decryption is possible). Therefore, we actually work with the weaker assumption that *homomorphic commitment* exists. Informally speaking, a homomorphic commitment scheme is a (semantically-secure, perfectly binding, non-interactive) commitment scheme C (cf., [8, Sec. 4.1.1]) that has the additional property that from commitments $C(x), C(x')$ it is possible to compute efficiently a commitment to $x + x'$, where $+$ is the operation of some group G.

Below, we formally define the notion of "homomorphic commitment scheme". We stress that this definition is *not* necessarily the most general definition that is possible here; instead, it is aimed at the simplicity of the presentation. Later in the paper, these results are strengthened in various ways.

Definition 2. *A* (semantically secure) homomorphic commitment scheme *consists of a (group-size) function* $L(k) : \mathbb{N} \to \mathbb{N}$ *and a triplet of algorithms* $(\textsc{Gen}, \textsc{Commit}, \textsc{Add})$ *as follows.*

1. $\textsc{Gen}$ *is a probabilistic polynomial-time key-generation algorithm; on input* 1^k *it outputs a public-key* PK.
2. *The commitment algorithm* $\textsc{Commit}$ *is a probabilistic polynomial-time algorithm that takes input* 1^k*, the public-key* PK *and a string* x *which is an element of the group* $Z_{L(k)}$ *(where* $L(k)$ *is a prime); it outputs a string* $\textsc{Commit}_{\text{PK}}(x)$ *of some length* $p(k)$*. On one hand, this string* hides *the value* x*; i.e., given* $\textsc{Commit}_{\text{PK}}(x)$*, the value* x *is semantically secure. (Note that the notation* $\textsc{Commit}_{\text{PK}}(x)$ *hides the fact that the algorithm* $\textsc{Commit}$ *is probabilistic. When we wish to emphasize this fact, we sometimes use the notation* $\textsc{Commit}_{\text{PK}}(x, \cdot)$*. In other cases, we may wish to obtain a deterministic value by fixing some randomness* r *to the algorithm* $\textsc{Commit}$*; in such a case we use the notation* $\textsc{Commit}_{\text{PK}}(x, r)$*.) On the other hand, the commitment is* perfectly binding*; i.e., given* PK*, the commitment string* $\textsc{Commit}_{\text{PK}}(x)$ *uniquely determines* x*.*[2]
3. *The composition algorithm* $\textsc{Add}$ *is a probabilistic polynomial-time algorithm that takes input* 1^k*, the public-key* PK *and two commitments* $\textsc{Commit}_{\text{PK}}(x)$*,* $\textsc{Commit}_{\text{PK}}(x')$ *and computes a commitment to* $x + x'$*; i.e.,* $\textsc{Commit}_{\text{PK}}(x + x', r)$*, where* $+$ *refers to addition operation in the group* $Z_{L(k)}$ *and* r *is any possible randomness for the commitment algorithm,* $\textsc{Commit}$*. Finally, we require that commitments can be* re-randomized.[3] *That is, there is a probabilistic polynomial-time algorithm* $\textsc{ReRand}$ *that, given any commitment* $\textsc{Commit}_{\text{PK}}(x, r)$*, outputs a re-randomized commitment distributed according to* $\textsc{Commit}_{\text{PK}}(x, \cdot)$ *(of the same string* x*).*

Example 2. A simple example is the quadratic-residuosity based probabilistic encryption of [11]; in this case the group that is used is Z_2. For an additional example, consider the ElGamal commitment: Let p be a prime, and let g be a generator of a subgroup $G \subseteq Z_p^*$ of prime order q in which the discrete-log problem is "hard". Let $\text{PK} = (p, q, g, g^a)$, for some a. The commitment is defined by $\textsc{Commit}_{\text{PK}}(x, b) = (g^b, g^x \cdot g^{ab})$. Note that by taking the product of two commitments, i.e. $\textsc{Commit}_{\text{PK}}(x, b) \odot \textsc{Commit}_{\text{PK}}(x', b')$, we get

$$(g^b, g^x \cdot g^{ab}) \odot (g^{b'}, g^{x'} \cdot g^{ab'}) = (g^{b+b'}, g^{x+x'} \cdot g^{a(b+b')}) = \textsc{Commit}_{\text{PK}}(x+x', b+b').$$

Also note that if we work directly with ElGamal encryption (i.e., with x instead of g^x) then this allows decryption, but the product gives a value that corresponds to $x \cdot x'$ rather than to $x + x'$.

[2] i.e., for all $(x, r), (x', r')$ such that $x \neq x'$, we have $\textsc{Commit}_{\text{PK}}(x, r) \neq \textsc{Commit}_{\text{PK}}(x', r')$; this is the analogue of the (perfect) correctness property in the terminology of *encryption*.

[3] This requirement, which is standard for most applications of homomorphic encryption, is actually not used in this section but will be needed in Section 5.

Remark 2. Observe that the definition guarantees also that, for any integer $c \geq 0$, a commitment to cx (i.e., a value $\text{COMMIT}_{\text{PK}}(cx, r)$ for some r) can be efficiently computed (using repeated doubling) from $\text{COMMIT}_{\text{PK}}(x)$ by applying the algorithm ADD $O(\log_2 c)$ times.

Construction: Given an arbitrary homomorphic commitment scheme, i.e. a triplet (GEN, COMMIT, ADD), we construct a CRHF family as follows. The key-generation algorithm of the hash-family GEN′, on input 1^k, works by first applying $\text{GEN}(1^k)$ to obtain a public-key PK and then choosing at random an $n_1 \times n_2$ matrix M (where n_1, n_2 are specified below) whose elements are in $Z_{L(k)}$. The index for the hash-function that GEN′ outputs is $s = (\text{PK}, \text{COMMIT}_{\text{PK}}(M))$, where $\text{COMMIT}_{\text{PK}}(M)$ consists of commitments to each of the $n_1 \cdot n_2$ elements of the matrix M. The function H_s, on input $x = (x_1, \ldots, x_{n_2})$ (where each x_i is $l(k)$-bit string and $l(k) = \lfloor \log_2 L(k) \rfloor$), is defined as follows:

$$H_{\text{PK},\text{COMMIT}_{\text{PK}}(M)}(x) \stackrel{\text{def}}{=} \text{COMMIT}_{\text{PK}}(M \cdot x, r),$$

where the commitment to $M \cdot x$ can be efficiently computed from s and x using ADD and Remark 2 above. (Here r is the randomness implicitly defined by this computation.) It remains to prove that collisions are hard to find. Assume towards a contradiction, that there exists an algorithm B that, given s, finds (with high probability) a pair x, x' that forms a collision for H_s; i.e., $H_s(x) = H_s(x')$ or alternatively $\text{COMMIT}_{\text{PK}}(M \cdot x, r) = \text{COMMIT}_{\text{PK}}(M \cdot x', r')$. It follows, by the perfect binding property, that $M \cdot x = M \cdot x'$ or that $M(x - x') = 0$. This contradicts the semantic security of COMMIT, as we found a vector y in the kernel of the committed matrix M. According to the semantic security, this should have been possible only with probability which is very close to the a-priori probability; if we choose $n_1 = \lceil k/l(k) \rceil$ then this a-priori probability is at most $1/2^k$. [4] Finally, the parameter n_2 is chosen such that the output of H_s (whose length is $n_1 \cdot p(k)$) is shorter than its input (whose length is $n_2 \cdot \log_2 L(k)$).

We summarize the above discussion with the following theorem:

Theorem 1. *If there exists a homomorphic commitment scheme then there exists a family of CRHFs.*

To conclude this section, we would like to offer (in an informal manner) a slightly more general view of the above construction. Assume that we are given homomorphic commitment scheme, as above, and in addition a *linear MAC*. We construct a family of CRHFs as follows. The index of each function s consists of a public-key for the commitment, PK, and an "encryption" (by applying $\text{COMMIT}_{\text{PK}}$) of a MAC-key SK. The function $H_s(x)$ is defined by

$$H_s(x) \stackrel{\text{def}}{=} \text{COMMIT}_{\text{PK}}(\text{MAC}(x), r).$$

[4] Note that if the group-size is sufficiently large then n_1 might be as small as 1.

As before, computing the commitment to $\text{MAC}(x)$ can be done (without knowing the secret-keys) based on the linearity of the MAC and the homomorphic properties of the commitment. Now, assume that an adversary can efficiently come up with a collision x, x' such that $\text{COMMIT}_{\text{PK}}(\text{MAC}(x), r) = \text{COMMIT}_{\text{PK}}(\text{MAC}(x'), r')$ (for some r, r'). Again, by the perfect binding, it follows that $\text{MAC}(x) = \text{MAC}(x')$ which contradicts the security of the MAC. Hence, if the MAC is secure then the only other possibility is that the adversary, by examining s, could obtain information about the MAC secret-key; this, in turn, contradicts the security of COMMIT (which is used to "encrypt" this key).

4 CRHF from PIR

In this section, we show a construction of CRHFs based on (computationally) private information retrieval (PIR) schemes. (In fact, our construction can also use PIR schemes where the user's reconstruction is unbounded.) Since PIR is implied by homomorphic encryption [15, 21, 16] (and unbounded PIR by homomorphic commitment), this result is stronger than the result presented in Section 3. The nature of the construction presented in this section is combinatorial, as opposed to the algebraic nature of the constructions presented in Section 3 and Section 5.

Definition 3. *A* (computational, 1-round) PIR scheme *is a protocol for two parties: a* user*, $\mathcal{U}$, and a* server*, $\mathcal{S}$. The server holds a database $x \in \{0,1\}^n$ and the user holds an input $i \in [n]$. The goal of a PIR scheme is for the user to learn the value of the bit x_i while keeping the value of i hidden from the server. The protocol uses only one round of interaction: $\mathcal{U}$ sends to the server a* query*, $q = \text{QUERY}(1^n, i, \rho)$, where ρ is the user's random input, and it gets in return an answer $a = \text{ANS}(x, q)$. The user then applies a reconstruction algorithm* REC *to compute $x_i = \text{REC}(a, i, \rho)$. The 3 algorithms $(\text{QUERY}(\cdot), \text{ANS}(\cdot), \text{REC}(\cdot))$ that define the PIR scheme are polynomial-time algorithms that should satisfy the following two requirements:*

1. *(Correctness) The user always retrieve x_i correctly (where the probability is over the choice of the user's random input ρ).*
2. *(Privacy) For every two indices $i, j \in [n]$ the corresponding distributions of queries, $\text{QUERY}(1^n, i, \cdot)$ and $\text{QUERY}(1^n, j, \cdot)$, are indistinguishable. (Alternatively, it will be useful to talk about* semantic security *of the query rather than about indistinguishability; namely, no adversary can gain a significant advantage in guessing a predicate of i given $q = \text{QUERY}(1^n, i, \cdot)$.)*

The main complexity measure for PIR schemes is their communication complexity. Specifically, we denote by $\alpha(n)$ the (worst-case) query length (over all $x \in \{0,1\}^n$ and all possible choices of ρ) and by $\beta(n)$ the (worst-case) answer length.

It is easy to see that any one-round PIR query to a random address is already a collision-resistant hash function *in itself*, since any adversary who finds

collisions contradicts semantic security of PIR query. We show a hash function with somewhat tighter security reduction, using an error correcting code. Specifically, we will use any error correcting code $\text{ECC}(\cdot)$ that expands $x \in \{0,1\}^k$ to $y \in \{0,1\}^n$, where $n = c \cdot k$ (for a constant c) and that can correct up-to $\lambda \cdot n$ errors (for a constant λ).[5]

Construction: Given a PIR scheme (QUERY, ANS, REC) and an error correcting code ECC, we construct a CRHF family as follows. The key-generation algorithm of the hash-family, GEN, on input 1^k, works by choosing $t = \omega(\log k)$ queries $q_1, \ldots, q_t$ (this is done by choosing t random strings $\rho_1, \ldots, \rho_t$ and t random indices $i_1, \ldots, i_t \in_R [n]$, where as above $n = c \cdot k$, and computing $q_j = \text{QUERY}(1^n, i_j, \rho_j)$). The hash-index is $s = (q_1, \ldots, q_t)$. The function H_s, on input $x \in \{0,1\}^k$, is defined as

$$H_s(x) = (\text{ANS}(y, q_1), \ldots, \text{ANS}(y, q_t)) \ ,$$

where $y = \text{ECC}(x)$. Clearly, H_s is computable in polynomial time. It maps strings of length k to strings of length $t \cdot \beta(n)$ (a possible choice of parameters is $t = \text{polylog}(k)$ and $\beta(n) = n^\epsilon = (ck)^\epsilon$; in such a case H_s indeed shrinks its input). Next, we argue that the resulting family is indeed collision-resistant. Suppose that an adversary can find a collision for H_s; i.e., it can find different strings x, x' such that $H_s(x) = H_s(x')$ or, equivalently, such that $(\text{ANS}(y, q_1), \ldots, \text{ANS}(y, q_t)) = (\text{ANS}(y', q_1), \ldots, \text{ANS}(y', q_t))$, where $y = \text{ECC}(x)$ and $y' = \text{ECC}(x')$. This implies (by the correctness of the PIR) that $y_{i_j} = y'_{i_j}$, for $1 \leq j \leq t$. However, since y, y' are distinct codewords of the error-correcting code then the distance between y, y' is at least $2\lambda n$; since each i_j is random, the probability that for a certain i_j we have $y_{i_j} = y'_{i_j}$ is constant (specifically, 2λ) and the probability that $y_{i_j} = y'_{i_j}$ for all j is (by the choice of t) negligible. By the semantic security of QUERY, finding such y, y' given q should be possible only with a negligible probability. This gives the desired contradiction.

Thus, we have:

Theorem 2. *If there exists a 1-round (single-server) PIR scheme with communication complexity $O(n^c)$ for some $c < 1$ then there exists a family of CRHFs.*

Remark 3. Fischlin [7] shows the impossibility of a black-box transformation from one-way trapdoor permutations to (one-round, computational) PIR. Our transformation from PIR to CRHF, together with the results of [20], yields a completely different way to obtain the same result.

5 CRHF from Homomorphic One-Way Commitment

The construction of CRHF from homomorphic encryption (or even from homomorphic commitment), presented in Section 3, seem to rely heavily on the

[5] It suffices for us that the encoding algorithm $\text{ECC}(\cdot)$ will work in polynomial time. It is not needed for us that the error correction will be efficient; we will only rely on the "large" distance between codewords.

semantic-security of the underlying commitment. In this section, we show that this is not really essential. Namely, we consider a primitive that we term *homomorphic one-way commitment*. In this case, the security of the committed value $\textsc{Commit}(x)$ does not guarantee that no information about x is leaked but only that it is hard (for a randomly chosen x) to "invert" the commitment and find x. Note however that it does not suffice to require that $\textsc{Commit}_{\mathrm{PK}}(x, r)$ is a one-way function, as we not only require that finding a pre-image (x, r) is hard but that even finding x alone is hard.

Definition 4. *A* one-way homomorphic commitment *is defined as homomorphic commitment (Definition 2), except for the security requirement:*

- *(One-Wayness) Every probabilistic polynomial-time algorithm I that is given $\textsc{Commit}_{\mathrm{PK}}(x, r)$ has a negligible probability of finding x, where the probability is over a random choice of $x \in Z_{L(k)}$, the choice of r by* Commit*, and the internal random choices of I. (Note that, by the binding property, for every value $\textsc{Commit}_{\mathrm{PK}}(x, r)$, there is a unique pre-image x.)*

Remark 4. The one-wayness requirement in particular implies that the size of the group from which x is taken, i.e. $L(k)$, needs to be "large". This is in contrast with the definition of "standard" homomorphic commitment where the group might be as small as Z_2. On the other hand, any (standard) homomorphic commitment where $L(k) = k^{\omega(1)}$ is immediately also a one-way homomorphic commitment. We can turn any standard homomorphic commitment (over an arbitrarily small group) into a one-way homomorphic commitment by concatenating "sufficiently many" copies of the original scheme (where $\omega(\log k)$ copies are always enough). Such a concatenation yields a group which is a product group, and in particular is not a cyclic group. It is possible to extend our results to such groups as well.

Construction: Given an arbitrary one-way homomorphic commitment scheme $(\textsc{Gen}, \textsc{Commit}, \textsc{Add})$, we construct a CRHF family as follows. The key-generation algorithm of the hash-family $\textsc{Gen}'$, on input 1^k, works by first applying $\textsc{Gen}(1^k)$ to obtain a public-key PK. Then, it chooses m random elements $x_1, \ldots, x_m \in_R Z_{L(k)}$ (where m, as before, is chosen so that the output length is shorter than the input length) and m random strings $r_1, \ldots, r_m$ to be used by the commitment algorithm. It finally computes m values $y_i = \textsc{Commit}_{\mathrm{PK}}(x_i, r_i)$. The index of the hash function that $\textsc{Gen}'$ outputs is $s = (\mathrm{PK}, y_1, \ldots, y_m)$. The function H_s, on input $\boldsymbol{a} = (a_1, \ldots, a_m)$ (where each a_i is an $l(k)$-bit integer and, as before, $l(k) = \lfloor \log_2 L(k) \rfloor$), is defined as follows:

$$H_{\mathrm{PK},y_1,\ldots,y_m}(\boldsymbol{a}) \stackrel{\mathrm{def}}{=} \textsc{Commit}_{\mathrm{PK}}(\sum_{i=1}^{m} a_i x_i, r),$$

where, as in the construction of Section 3, we observe that by using algorithm Add (and Remark 2) this commitment can be efficiently computed (without knowledge of $x_1, \ldots, x_m$) from s and $\boldsymbol{a}$ (and r is the corresponding randomness). It remains to prove that collisions are hard to find. Assume towards a

contradiction, that there exists an algorithm B that, given s, finds (with high probability) a pair $\boldsymbol{a}, \boldsymbol{a}'$ that forms a collision for H_s; i.e., $H_s(\boldsymbol{a}) = H_s(\boldsymbol{a}')$. This means that $\textsc{Commit}_{\text{PK}}(\sum_{i=1}^{m} a_i x_i, r) = \textsc{Commit}_{\text{PK}}(\sum_{i=1}^{m} a'_i x_i, r')$ which, by the perfect binding of the commitment, implies that $\sum_{i=1}^{m} a_i x_i = \sum_{i=1}^{m} a'_i x_i$. Therefore, the vector $\boldsymbol{d} = \boldsymbol{a} - \boldsymbol{a}'$ (which is easily computable from the collision) is such that $\boldsymbol{d} \cdot \boldsymbol{x} = 0$. We want to use the procedure that finds such vectors $\boldsymbol{d}$ in order to construct an inverter I for the commitment (i.e., an algorithm that finds $\boldsymbol{x}$ from $\textsc{Commit}(\boldsymbol{x})$, for a uniformly random $\boldsymbol{x}$) in contradiction to the one-wayness. While each such $\boldsymbol{d}$ gives some information about $\boldsymbol{x}$, applying the procedure repeatedly should be done with some care to avoid getting vectors $\boldsymbol{d}$ which are linearly dependent and are therefore useless. Next, we describe an inverter that uses the above ideas in a more careful way.

The inverter: The algorithm I (the inverter) that we construct gets as input a public-key PK and a vector $\boldsymbol{z}$ of m commitments (where PK, $\boldsymbol{z}$ and the randomness are all chosen at random with the appropriate distributions) and it finds, with non-negligible probability, the vector $\boldsymbol{x}$ of m committed values.[6] The inverter I repeats the following at most M times (where $M = O(m \cdot q(k))$ and $q(k)$ is the polynomial such that B succeeds with probability at least $1/q(k)$) or until I collects m linearly independent equations about $\boldsymbol{x}$. In the jth iteration, I picks at random an $m \times m$ matrix C_j and a length m vector $\boldsymbol{b}_j$. The elements of both C_j and b_j are taken from $Z_{L(k)}$ (and recall that we assume here that $L(k)$ is a prime number). Denote $\boldsymbol{x}_j = C_j \cdot \boldsymbol{x} + \boldsymbol{b}_j$ (of course the inverter does not compute this value as $\boldsymbol{x}$ is not available to it; we use $\boldsymbol{x}_j$ to simplify notation). The inverter computes $\boldsymbol{w}_j = \textsc{Commit}_{\text{PK}}(\boldsymbol{x}_j) = \textsc{Commit}_{\text{PK}}(C_j \cdot \boldsymbol{x} + \boldsymbol{b}_j)$ (that can be computed from $\boldsymbol{z}$ using the algorithm ADD) and finally it re-randomizes this vector of commitments (using the algorithm RERAND); i.e., it computes $\boldsymbol{y}_j = \textsc{ReRand}_{\text{PK}}(\boldsymbol{w}_j)$. The inverter provides $s_j = (\text{PK}, \boldsymbol{y}_j)$ to algorithm B and in return it gets a collision $\boldsymbol{a}_j, \boldsymbol{a}'_j$ for the hash function H_{s_j} (note that it is easy to check whether this pair is indeed a collision, simply by applying the function H_{s_j}; we can therefore assume that B itself either outputs a collision or the value "fail"). If B returns "fail" the current iteration is terminated and I proceeds to the next iteration; otherwise, I sets $\boldsymbol{d}_j = \boldsymbol{a}_j - \boldsymbol{a}'_j$ (and, as above, the vector $\boldsymbol{d}_j$ satisfies $\boldsymbol{d}_j \cdot \boldsymbol{x}_j = 0$). The iteration ends by computing the vector $\boldsymbol{u}_j = \boldsymbol{d}_j \cdot C_j$ and the scalar $\lambda_j = \boldsymbol{d}_j \cdot \boldsymbol{b}_j$. Note that $\boldsymbol{u}_j \cdot \boldsymbol{x} + \lambda_j = \boldsymbol{d}_j \cdot C_j \cdot \boldsymbol{x} + \boldsymbol{d}_j \cdot \boldsymbol{b}_j = \boldsymbol{d}_j(C_j \cdot \boldsymbol{x} + \boldsymbol{b}_j) = \boldsymbol{d}_j \cdot \boldsymbol{x}_j = 0$. Hence, if all goes well, the inverter ends the jth iteration with a new linear constraint about $\boldsymbol{x}$. Moreover, we will argue that after M iterations I is likely to have m linearly independent constraints; this allows solving the system of equations and to find $\boldsymbol{x}$; i.e., to invert the commitment.

It remains to prove that I succeeds in inverting the commitment $\boldsymbol{z}$ with non-negligible probability (assuming that B succeeds in finding collisions with non-negligible probability). For this, we make several simple observations. First, note that I invokes B several times, all with the same PK (which is part of

[6] Note that this is slightly stronger than what we need, since we "invert" m commitments at once; however we get this feature "for free".

all the indices s_j). We call a public-key PK *good* if B, when given a randomly chosen index of a hash function that includes this public-key (i.e., $s = (\text{PK}, \boldsymbol{y})$ for a randomly chosen $\boldsymbol{y}$) succeeds with non-negligible probability. Since B finds a collision with non-negligible probability over a randomly chosen s, it follows that a non-negligible fraction of the public-keys are good. Therefore, the probability that PK that is given to the inverter is good is non-negligible. From now on, we will assume that this is indeed the case. Fix any $\boldsymbol{z}$ (and hence also $\boldsymbol{x}$). Next, we note that, for every j, the vector $\boldsymbol{x}_j$ (whose choice is determined by the random choice of $C_j, \boldsymbol{b}_j$) is totally random and, moreover, these vectors are all independent. Hence, when B is given s_j (that includes PK and $\boldsymbol{y}_j$ – a rerandomized commitment to $\boldsymbol{x}_j$), by the assumption that PK is good, B succeeds in finding a collision with non-negligible probability. Finally, we argue that the vector $\boldsymbol{u}_j$ obtained from this collision is random; if this is true then indeed $O(m)$ successful iterations suffice (with high probability) and therefore total of M iterations are enough. This is so because $\boldsymbol{x}_j$ (and hence also the input for B; i.e., $\boldsymbol{y}_j$) is independent of C_j (because no matter what $\boldsymbol{x}, C_j$ are, the choice of $\boldsymbol{b}_j$ yields a uniformly random $\boldsymbol{x}_j$). Therefore $\boldsymbol{d}_j$ (the output of B) is also independent of the matrix C_j; hence, computing $\boldsymbol{u}_j = \boldsymbol{d}_j \cdot C_j$ (where $\boldsymbol{d}_j \neq \boldsymbol{0}$) yields a random vector.

To summarize, we state the following theorem:

Theorem 3. *If there exists a homomorphic one-way commitment scheme then there exists a family of CRHFs.*

Acknowledgements. We thank the anonymous referees for helpful comments and pointers.

References

1. B. Barak. How to Go Beyond the Black-Box Simulation Barrier. *Proc. of 42nd FOCS*, pp. 106–115, 2001.
2. B. Barak, and O. Goldreich. Universal Arguments and their Applications. *Proc. of 17th Conference on Computational Complexity*, pp. 194-203, 2002.
3. A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-Way Functions Are Essential for Single-Server Private Information Retrieval. *Proc. of 31st STOC*, pp. 89–98, 1999.
4. C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. *Proc. of IACR EUROCRYPT*, LNCS 1592, pp. 402–414, 1999.
5. I. Damgård: Collision Free Hash Functions and Public Key Signature Schemes. In *Proc. of EUROCRYPT*, pages 203-216, 1987.
6. I. Damgard, T. P. Pedersen and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In *Proc. of IACR Crypto*, LNCS 773, pp. 250–265, 1993.
7. M. Fischlin. On the Impossibility of Constructing Non-interactive Statistically-Secret Protocols from Any Trapdoor One-Way Function. *Proc. of CT-RSA*, pp. 79-95, 2002.

8. O. Goldreich. *Foundations of Cryptography. Volume I: Basic Tools*. Cambridge University Press, 2001.
9. O. Goldreich. *Foundations of Cryptography. Volume II: Basic Applications*. Cambridge University Press, 2004.
10. O. Goldreich, S. Goldwasser, and S. Halevi. Collision-Free Hashing from Lattice Problems. ECCC TR-42, 1996.
11. S. Goldwasser, and S. Micali. Probabilistic Encryption. *Journal of Computer and systems sciences* 28, 270-299, 1984.
12. S. Halevi, and S. Micali, Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In *Proc. of IACR Crypto*, LNCS 1109, pp. 201-215, 1996.
13. C.Y. Hsiao and L. Reyzin. Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins? In *Proc. of IACR Crypto*, 2004.
14. J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. *Proc. of 24th STOC*, pp. 723–732, 1992.
15. E. Kushilevitz and R. Ostrovsky. Replication is Not Needed: Single Database, Computationally-Private Information Retrieval. In *Proc. of 38th FOCS*, pages 364–373, 1997.
16. E. Mann. Private access to distributed information. Master's thesis, Technion – Israel Institute of Technology, Haifa, 1998.
17. S. Micali. CS Proofs. *SIAM J. Computing*, Vol. 30(4), pp. 1253-1298, 2000. (Early version appeared in FOCS 1994.)
18. W. Ogata, and K. Kurosawa. On Claw Free Families. IEICE Trans., Vol.E77-A(1), pp. 72-80, 1994. (Early version appeared in AsiaCrypt'91.)
19. A. Russell. Necessary and Sufficient Conditions for Collision-Free Hashing. *J. Cryptology*, Vol. 8(2), pages 87-100, 1995. (Early version in CRYPTO92).
20. D. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In *Proc. of EUROCRYPT*, pages 334-345, 1998.
21. J. P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 1998.
22. X. Wang, D. Feng, X. Lai, and H. Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive TR-199, 2004.