On the Design of Provably-Secure Cryptographic Hash Functions

Alfredo De Santis*
Dipartimento di Informatica ed Applicazioni
Università di Salerno
84081 Baronissi (Salerno), Italy

Moti Yung
IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598

(extended summary)

Abstract

Recently, formal complexity-theoretic treatment of cryptographic hash functions was suggested. Two primitives of Collision-free hash functions and Universal one-way hash function families have been defined. The primitives have numerous applications in secure information compression, since their security implies that finding collisions is computationally hard. Most notably, Naor and Yung have shown that the most secure signature scheme can be reduced to the existence of universal one-way hash (this, in turn, gives the first trapdoor-less provably secure signature scheme).

In this work, we first present reductions from various one-way function families to universal one-way hash functions. Our reductions are general and quite efficient and show how to base universal one-way hash functions on any of the known concrete candidates for one-way functions. We then show equivalences among various definitions of hardness for collision-free hash functions.

1 Introduction

Cryptographic Hash Functions are important tools in secure information compression and as building blocks for other cryptographic procedures. A hash function is cryptographically strong if collision finding is computationally hard.

The usefulness of cryptographic hashing was used and known in practice [M1, Gi, M2] (and is already mentioned in the original Diffie-Hellman paper which introduced the notions of trapdoor and one-way functions and their applications). Nevertheless, only recently two formal complexity-theoretic definitions of cryptographic hash functions were given and implementations based on one-way functions were suggested.

*Part of this work was done while the author was visiting IBM Research Division, T. J. Watson Research Ctr, Yorktown Heights, NY 10598.

I.B. Damgard (Ed.): Advances in Cryptology - EUROCRYPT '90, LNCS 473, pp. 412-431, 1991.

The first function family, suggested by Damgård, is the *Collision-Free Hash Functions* (CFHF) [D], which is based on *Claw-Free* functions of Goldwasser, Micali and Rivest [GMRi]. The second function family, suggested by Naor and Yung [NY], is the *Universal One-Way Hash Function* family (UOWHF). In a CFHF family, the function is given and then finding a colliding pair is hard, while in UOWHF definition is weaker: first an adversary chooses an input to compress and then the function is drawn at random; (the weaker definition may imply implementations based on weaker assumptions).

CFHF family can be based on any one-way homomorphism. Its applicability was shown as this family when having also a trapdoor property was used to implement a secure signature scheme [GMRi, D]; it was also shown to give efficient zero-knowledge proof-systems [NY]. On the other hand, [NY] showed that a secure signature scheme is reduces to the existence of UOWHF, and they show how to achieve UOWHF based on 1–1 one-way functions, giving the first provably secure signature scheme which is trapdoor-less (unlike all previous secure signature schemes which had followed the Diffie-Hellman model of basing signatures on a trapdoor property).

It is theoretically important to base cryptographic primitives and basic tools on reduced complexity assumptions, it is also practically important to give efficient implementations of such tools. In this work we investigate implementations of cryptographic hash functions on reduced complexity assumptions and investigate reductions among various definitions of hardness of collision finding. We would like the reductions to be efficient as well.

We first give a reduction from a 1–1 one-way function to a UOWHF (our proof is easier and the reduction is more efficient than the original construction in [NY]). We then show that a UOWHF can be based on a one-way function with the property that the expected size of the preimage of an element in the range is small (i.e., when an element in the domain is randomly chosen). We call this function *small expected preimage-size function*. We then show how to construct such a function if a regular function [GKL] is available; this function family includes a large number of concrete examples (see [GKL]). Even more generally, we show how to reduce a very general one-way function family to a small expected preimage-size family. The general property requirement is that: given an element in the range, an estimate on the size of the preimage set is almost always easily computable (where the estimate should only be polynomially close to the real size). We call such a function an *almost-known preimage-size function*. This requirement is a mild one since it implies some structure of the domain-range relationship which all concrete candidates for one-way functions (based on number theory, algebra, coding theory or combinatorics (subset-sum)) have. Then, we investigate various definitions of hardness of hash functions and we show reductions and equivalences among the various definitions; such relations may lead to finding CFHF based on reduced complexity assumptions or on concrete functions which are assumed to be one-way.

Recently, Rompel has come up with a construction of generating a UOWHF based on any one-way function [Ro]. This general construction does not rely on the approximate knowledge of the preimage size (the property mentioned above) and it is much more involved than ours. It is ingenious and theoretically

optimal, as it shows that the Naor-Yung approach leads to a signature based on any one-way function (which is necessary by the work of Impagliazzo and Luby [IL]). However it is much less practical than the work presented here for all known concrete candidates for one-way functions.

The rest of the paper is organized as following. Section 2 present background on one-way functions, universal hash functions, UOWHF, and signature schemes. The reader familiar with these notions can skip this section. Section 3 gives the construction based on any 1-1 one-way function, while Section 4 gives the construction based on small expected preimage-size, regular, and the most general almost-known preimage-size families. In section 5 we present reductions among different notions of difficulty of cryptographic hash functions.

2 Notations and background

Let x be a string, then $|x|$ is the length of x. Let x and y be two strings, then $x \diamond y$ is the concatenation of x and y. By the symbol "$\circ$" we mean the composition of functions. Thus, let f and g be functions, $y = f \circ g(x)$ is the value $f(g(x))$.

Probability ensembles: A probability ensemble D is the set $\{D_n \mid n \in N^+\}$ where D_n is a distribution probability on $\{0,1\}^n$. For $x \in \{0,1\}^n$, $D[x]$ is the probability assigned to x by D_n. For $X \subseteq \{0,1\}^n$, $D[X]$ is the sum $\sum_{x \in X} D[x]$. By the notation $x \in_R B$, we mean that x has been chosen from the set B under the uniform distribution (i.e. each element in B has the same probability $1/|B|$ of being selected).

Accessible ensembles: An ensemble D is accessible if there exists a probabilistic polynomial time algorithm G, such that on input n, the probability distribution induced by the output of G (depending on its internal coin-flips) is D_n.

Functions: A function f is a collection $\{f_n\colon \{0,1\}^n \rightarrow \{0,1\}^{l(n)} \mid n \in N^+\}$ where $l(n)$ is the output length. Hereafter, for sake of brevity we often omit the subscript in f_n. All functions considered will be polynomial time computable, i.e. given an input n and an argument x, the value $f_n(x)$ can be computed in time polynomial in n.

Definition 1 [One-way function.] *f is one-way if for every polynomial time algorithm A, for all polynomials p and all sufficiently large n, the probability that A on input $f(x)$, when $x \in_R \{0,1\}^n$, outputs a y such that $f(y) = f(x)$ is*

$$Pr[f(x) = f(A(f(x))) \mid x \in_R \{0,1\}^n] < 1/p(n).$$

We remark that the above definition is of a *strong one-way function* which is implied by the existence of the weaker *somewhat one-way function* using Yao's amplification technique [Y]. A somewhat one-way function has the same definition as above, but the hardness of inversion is smaller, i.e. its probability is inverse polynomially away from 1. (In the above definition the probability is at most $1 - 1/q(n)$ for a given polynomial (instead of $1/p(n)$ for any polynomial above)).

Unless stated otherwise, f will have input length n and output length $l(n)$. The requirement that the range of f has a uniform length is without loss of generality, as we may use a variable to fixed length encoding. For instance, if f has output length less than or equal to m, then we can construct a function f' with output length at most $2m$ (or $m + 2\lceil \log m \rceil + 2$) by employing a suitably prefix encoding [E] explained below.

A prefix set of strings is a set S with the property that if any two strings $x, y \in S$ are such that $x = y \diamond w$ (where w is a string) then $x = y$ (and w is the empty string). A prefix encoding from a set S to a set R is a bijection from S to R where R is a prefix set.

In our case, the set is the range of a one-way function f (where $f = f_n$), we employ the following encoding $f'(x) = 0^{|f(x)|-1} \diamond 1 \diamond f(x)$. This has the property that it is easy to compute $f'(x)$ given $f(x)$ and vice-versa, even without knowing x. Since the range of f' is a prefix set, we can add dummy zeroes to make the range of the same length m. That is $f''(x) = f'(x) \diamond 0^{m-|f'(x)|}$.

2.1 Collision-Free Hash Functions

Let $\{n_{1_i}\}$ and $\{n_{0_i}\}$ be two increasing sequences such that for all i $n_{0_i} \leq n_{1_i}$, but $\exists q$, a polynomial such that $q(n_{0_i}) \geq n_{1_i}$ (we say that these sequences are polynomially related). Let H_k be a collection of functions such that for all $h \in H_k$, h: $\{0,1\}^{n_{1_k}} \mapsto \{0,1\}^{n_{0_k}}$ and let $U = \bigcup_k H_k$. Let A be a probabilistic polynomial time algorithm (A is a *collision adversary*) that given a random $h \in H_k$ attempts to find $x, y \in \{0,1\}^{n_{1_k}}$ such that $h(x) = h(y)$ but $x \neq y$. In other words, after getting a hash function it tries to find a collision pair.

Definition 2 *Such a U is called a* family of Collision-Free Hash Functions *(CFHF) if for all polynomials p, for all polynomial time probabilistic algorithms A, and all sufficiently large k the following holds:*

1. *$Pr[A(h) = (x,y), h(x) = h(y), y \neq x] < 1/p(n_{1_k})$ where the probability is taken over all $h \in H_k$ and the random choices of A.*
2. *$\forall h \in H_k$ there is a description of h of length polynomial in n_{1_k}, such that given h's description and x, $h(x)$ is computable in polynomial time.*
3. *H_k is accessible : there exists an algorithm G such that G on input k generates uniformly at random a description of $h \in H_k$.*

Based on the existence of claw-free permutations (as defined in [GMRi]) one can construct a CFHF [D]; also, based on any one-way function which is homomorphism, one can construct CFHF.

2.2 Universal Hash Functions

The following definition is from Carter and Wegman [CW].

Definition 3 [Universal$_2$ hash function.] *Let G be a family of functions from C to B. We say that G is a* universal$_2$ *if for any pair of inputs (a_1, a_2) and any pair of outputs (b_1, b_2), the number of functions that map a_1 to b_1 and a_2 to b_2 is $|G|/|B|^2$.*

Let $x \in C$, $S \subseteq C - \{x\}$, $g \in G$ and $\delta_g(x,S)$ be the number of $y \in S$ such that $g(x) = g(y)$. Then by [CW], the expected value μ of $\delta_g(x,S)$, for each fixed x and S and when g is uniformly chosen from G, is $\mu = |S|/|B|$. Markov's inequality tells us that when g is randomly chosen from G then for any $t > 1$:

$$Pr[\delta_g(x,S) > t \cdot \mu] < 1/t.$$

Definition 4 [NY] (extended): *A strongly universal$_2$ family G has the* collision accessibility *property if, given a requirement $g(x) = a_1$ and $g(y) = a_2$, it is possible to generate in polynomial time a function uniformly among all functions in G that obey the requirement.*

The above is an extended property which is necessary for our construction. A simple example of such a family (which we will use from now on, just for clarity) is the set $G_{n,m} = \{g_{a,b} \mid g_{a,b}(x) = chop(ax+b), a,b \in GF(2^n)\}$, where all computations are in $GF(2^n)$ and $chop : \{0,1\}^n \rightarrow \{0,1\}^m$ returns the first m bits of its n-bit argument. We will denote by G_n the set $G_{n,n-1}$. An interesting property of the family $G_{n,m}$ is that each function is a 2^{n-m}-1 function; i.e. exactly 2^{n-m} elements in the domain have the same value in the range. In particular, when $n = m+1$ the functions are 2-1; and when $m = n$, $G_{n,n}$ is a permutation.

From now on, all the strong universal$_2$ families we consider are supposed to have the collision accessibility property.

The following simple lemma states that the composition of two universal$_2$ functions is still universal$_2$.

Lemma 1 [Composition.] *Let G_1 and G_2 be two universal$_2$ families from C_1 to C_2 and from C_2 to C_3, respectively. Then the set $G = \{g = g_2 \circ g_1 \mid g_1 \in G_1, g_2 \in G_2\}$ is a universal$_2$ family from C_1 to C_3.*

2.3 Universal One-way Hash Functions

In this subsection we review the definition and the important properties of universal one-way hash functions (UOWHF), as introduced and discussed in [NY].

Let $\{n_{1_i}\}$ and $\{n_{0_i}\}$ be two increasing sequences such that for all i $n_{0_i} \leq n_{1_i}$, but $\exists q$, a polynomial such that $q(n_{0_i}) \geq n_{1_i}$ (we say that these sequences are polynomially related). Let H_k be a collection of functions such that for all $h \in H_k$, h: $\{0,1\}^{n_{1_k}} \mapsto \{0,1\}^{n_{0_k}}$ and let $U = \bigcup_k H_k$. Let A be a probabilistic polynomial time algorithm (A is a *collision adversary*) that on input k outputs $x \in \{0,1\}^{n_{1_k}}$ which we call an *initial value*, then given a random $h \in H_k$ attempts to find $y \in \{0,1\}^{n_{1_k}}$ such that $h(x) = h(y)$ but $x \neq y$. In other words, after getting a hash function it tries to find a collision with the initial value.

Definition 5 *Such a U is called a* family of universal one-way hash functions *if for all polynomials p, for all polynomial time probabilistic algorithms A, and for all sufficiently large k the following holds:*

1. *If* $x \in \{0,1\}^{n_{1_k}}$ *is A's initial value, then* $Pr[A(h,x) = y, h(x) = h(y), y \neq x] < 1/p(n_{1_k})$ *where the probability is taken over all* $h \in H_k$ *and the random choices of A.*
2. $\forall h \in H_k$ *there is a description of h of length polynomial in* n_{1_k}*, such that given h's description and x,* $h(x)$ *is computable in polynomial time.*
3. H_k *is accessible : there exists an algorithm G such that G on input k generates uniformly at random a description of* $h \in H_k$*.*

Notice that H_k is actually a collection of descriptions of functions; two different descriptions might correspond to the same function.

In this definition the collision adversary A is a (uniform) algorithm. We can alternatively define UOWHF where A is a polynomial sized circuit (the non-uniform case). In this case, all our results still hold, but we require the one-way functions that we use to be one-way in the non-uniform setting as well.

An important property is the composition lemma: composing families of UOWHF yields a family of UOWHF. Because of this lemma it will be enough to prove the existence of UOWHF that compress one bit. Invoking the composition lemma allows us to construct a family for any input and output size that are polynomially related.

Let $H_1, H_2, ..., H_l$ be families of functions such that $\forall i$ and $\forall h_i \in H_i$, h_i: $\{0,1\}^{n_i} \mapsto \{0,1\}^{n_{i-1}}$ and $n_i < n_{i+1}$. We call $H = \{h \mid h = h_1 \circ h_2 \circ ... \circ h_l\}$ an *l-composition* of $H_1, H_2, ..., H_l$. H is a multiset; If $h_1 \circ h_2 \circ ... \circ h_l = h'_1 \circ h'_2 \circ ... \circ h'_l$ for different $(h_1, h_2, \ldots, h_l)$ and $(h'_1, h'_2, \ldots, h'_l)$, both instances are members of H. (In other words, we use the set of concatenated functions and sample an element by sampling each H_i independently and uniformly).

Lemma 2 [NY] *Let H be an l-decomposition as above. If there exists an algorithm A that produces an initial value x and when given a uniformly random* $h \in H$ $Pr[A(h,x) = y, h(x) = h(y), y \neq x] > \epsilon$*, then there exists an* $1 \leq i \leq l$ *and an algorithm A' such that*

- *A' produces an initial value* $x_i \in \{0,1\}^{n_i}$
- *then on input* $h_i \in H_i$ *tries to find a* y_i *that collides with* x_i*.*
- $Pr[A'(h_i, x_i) = y_i, h(x_i) = h(y_i), y_i \neq x_i] > \epsilon/l$ *where the probabilities are taken over* $h_i \in H_i$ *and A's random choices.*
- *The running time of A' is similar to that of A.*

We remark that an equivalent definition to the above is when the initial input x is chosen (in a more specific way) at random. For a given $h \in H$ and x chosen by an arbitrary way by A, one can come up with another UOWHF family $H' = G_{n,n} \circ H$ where $G_{n,n}$ is a *universal*$_2$ permutation family, which randomizes the initial value.

UOWHF can be successfully applied to solve various authentications problems. Signature schemes and public fingerprintings for files among the others [NY]. Next we briefly describe signature schemes, and how to base a trapdoorless secure signature scheme on UOWHF.

2.4 Signature Schemes

In this subsection we review the definition of a signature scheme, its security and the relation of trapdoor-less signature schemes and UOWHF. For a more complete treatment, the reader is encouraged to consult the original paper [NY].

Digital signature is a primitive suggested right at the birth of modern (public-key) cryptography by Diffie and Hellman [DH]. The first implementations of their idea provided digital signature as well [MH, RSA, R]; these proposed signature schemes were based on trapdoor one-way functions, but lacked a precise notion of security. Following [DH], signature systems design has become an extensive field of research (see [GMRi]); we concentrate here only on provably secure systems.

The first scheme to deal formally with the notion of security of signature scheme was suggested by Goldwasser, Micali and Yao [GMY] who also pointed out flaws in the Diffie-Hellman scheme. They based their probabilistic scheme on the problem of factoring. Then, the strongest known definition of security was formalized by Goldwasser, Micali, and Rivest [GMRi]; they defined what it means for a system to be *existentially unforgeable under an adaptive chosen plaintext attack* (which we call "secure" in the rest of the paper). This is an attack by an adversary (forger) who initially computes a plaintext and receives from the signature algorithm a corresponding valid signatures; this is repeated in an adaptive fashion, polynomially many times. Then, the forger has to produce, without the cooperation of the signature algorithm, an extra signature for a message that was not previously signed. A secure system was designed under the assumption that factoring is hard, or a more general assumption that claw-free trapdoor permutations exist. Bellare and Micali [BeM] have shown how to construct secure signature system based on the assumption that trapdoor one-way permutations exist; this matches the original suggestion of Diffie and Hellman, but this time the system had a proof of security.

Naor and Yung [NY] were the first to conceive that the trapdoor property is not necessary for secure signature, (even one robust against the adaptive chosen plaintext attack). They proved that a one-way permutation is sufficient and invented the primitive of universal one-way hash family (UOWHF) to achieve (among other things) a secure signature.

2.4.1 Definition of a Signature Scheme and its Security

A signature scheme includes the following components:

1. A *security parameter* k which determines the size of keys, messages and other resources; all sizes and algorithms are polynomial in k.
2. A *message space* MS, we allow all messages of a given size polynomial in k.
3. A *key component* which includes a *key space* $KS(k)$ a family from which keys are being drawn and a *generation algorithm* KAL which chooses random keys.
4. A *signature bound* SB, a polynomial representing a bound on the number of messages signed; any polynomial should work.

5. A *system state* s which represents the state of the system; there is an initial state and execution states.
6. A *signing algorithm* SAL which is given a message, a system state, and a key, generates a signature and updates the system's state.
7. A *verification algorithm* VAL which is given a message, a signature and a system's state, checks the validity of the signature.

A signature system is a distributed system in which each user is a polynomial-time machine which initiates its instance of the signature scheme.

Next we describe attacks on signature schemes. The most general attack on a signature scheme ([GMRi]) has two phases. First, it allows a polynomial-time adversary F (a *forger*) to use the signature algorithm in an adaptive fashion, getting signatures to polynomially many plaintexts of its choice. Next, the attack has an existential nature, i.e., the forger itself has to come with a valid signature of a new message of its choice, in which case we say that it was successful. A scheme is *p-forgeable* if for a polynomial p there is a forger F which for infinitely many k's, succeeds in the attack with probability larger than $1/p(k)$, where the probability is taken over the random choices of keys by KAL, the choices of the signatures by SAL, and the coin flips of F itself. We say that a system is *secure* if it is not p-forgeable for any polynomial p.

2.4.2 A Signature Scheme based on UOWHF

Here we review the new approach to signature scheme as developed in [NY]. We briefly describe their reduction of signature to UOWHF.

Consider the Diffie-Lamport *tagging system* [La]. It consists of making public a one-way function f and *a window*, which is an ordered pair of values $< f(x_o), f(x_1) >$, for randomly chosen x_o, x_1 in the function domain. The user, then, is committed to the window and later on when it sends a bit b, it is done by publishing a tag x_b, an operation we call *opening half a window*. We say that the other half of the window remains *unused*. The construction can be extended to tag a message of length m-bits, by initially publishing (committing) to a *row of windows* $[< f(x_o^i), f(x_1^i) >, i = 1, ..., m]$ and then opening the halves corresponding to the bits of the message. Since f is one-way, only the committed user can open a tag, and no one else can tag a different message unless it can invert a random value of f, furthermore, anyone can verify tags; in this sense the system resembles a signature scheme.

The drawback of the above system is that the size of the initial commitment limits the number of bits which can be tagged; to eliminate it (and transform it into a signature scheme) [NY] suggests to use UOWHF. The general strategy of their system is to extend the tagging system, enhancing it with the capability of "regenerating rows of windows". The system is represented as a linked list, a system's state is a list consisting of nodes. Each node is associated with a message, i.e., it tags that message. The node is also connected to its successor in the list, i.e., it tags the successor node as well.

The node N_i contains three data fields: h_i a UOWHF, and two rows of windows rm_i and rs_i, the first will tag the next message M_{i+1} while the second

one will tag the successor node in the list, N_{i+1}. The UOWHF family is publicly known, or is otherwise produced by each user.

Next we sketch the algorithms and the dynamic behavior of the system. The system has an initial state (state 0) in which a user deposits an initial (root) node N_0 in the public directory.

In a typical situation the system is in state s_{i-1} where there is a list of $i-1$ nodes and the *last-node* N_{i-1} is unused. The connection between nodes will be explained in the following sketch of the signature and the verification algorithms.

SAL: Each message signing changes the state of the system, the list is grown by a node which becomes the new last-node. At state s_{i-1} the user sends a message M_i and tags it using the row rm_{i-1}, furthermore a new node N_i is generated by algorithm KAL: $N_i =< h_i, rm_i, rs_i >$ where its components are chosen at random: h_i is a random element of the UOWHF family based on f, and the rows are encryption by f of random tag values.

In order to link the new node into the list, the user has to tag the new node by its predecessor. Notice that the new node as a string of random bits is larger than the tagging capabilities of the row rs_{i-1} which was given this tagging task! Here is where the UOWHF hash is needed in a non-trivial way. The algorithm first computes the hash value of the new node by evaluating $n_i = h_{i-1}(N_i)$, then the smaller string n_i is being tagged by opening the corresponding half-windows in rs_{i-1}. This defines a signature on M_i and a new valid state of the system s_i.

VAL: A verification of a validity of a message can be done by checking the tagging of the message M_i by rm_{i-1} and testing the validity of the system's state by checking that the tagging of $n_j = h_{j-1}(N_j)$ is a valid one, namely, it was done by a proper opening of rs_{j-1} for all $j = 1, \ldots, i-1$. This is done all the way to the root and if all checks are valid the user accepts the signature.

Since a UOWHF implies the existence of a one-way function [NY], we can state that:

Theorem 1 *[NY] If UOWHF exist, then the signature scheme described above is secure.*

It is also possible to improve the efficiency of the above scheme [Go, NY].

3 UOWHF Based on 1–1 One-way Functions

Naor and Yung [NY] showed how to construct UOWHF from any 1-1 one-way function. In this section we describe a construction different from them which is easier to prove and is more economical in the number of applications of one-way function used in the construction. Actually only a single application of a one-way function is used.

Let f be a 1–1 one-way function (i.e. a 1–1 function that is also one-way), with input length n and output length $l(n)$.

Define $H_n = \{h = g_n \circ g_{n+1} \circ \ldots \circ g_{l(n)} \circ f_n \mid g_i \in G_i\}$, where G_i is a strongly universal$_2$ family from i-bit strings to $(i-1)$-bit strings.

Theorem 2 *$U = \bigcup_n H_n$ (based on 1-1 one-way function) is a UOWHF family.*

Proof. The proof is by contradiction. If there is a polynomial time algorithm that can find collisions for a randomly chosen h, then it can be used to invert the one-way function f.

Suppose there is an algorithm A, that on input n produces an $x \in \{0,1\}^n$ and given a randomly chosen $h \in_R H_n$ outputs a y such that $h(x) = h(y)$ and $x \neq y$, with probability greater than ϵ (here the probability is taken over $h \in_R H_n$, and the random choices of A). Given x and $g_n,\ldots, g_{l(n)}$, denote by COL_i, $n \leq i \leq l(n)$, the set of y such that $g_i \circ g_{i+1} \circ \ldots \circ g_{l(n)} \circ f_n(x) = g_i \circ g_{i+1} \circ \ldots \circ g_{l(n)} \circ f_n(y)$ and, for $i < l(n)$, $g_{i+1} \circ \ldots \circ g_{l(n)} \circ f_n(x) \neq g_{i+1} \circ \ldots \circ g_{l(n)} \circ f_n(y)$.

Let j, $n \leq j \leq l(n)$, be an integer such that with probability at least $\epsilon/(l(n) - n + 1)$ the algorithm A, on input x and $h \in_R H_n$, gives a value $A(x,h) \in COL_j$. Such a j must exist by the pigeonhole principle (wlog now we assume it is known).

Given x and $g_{j+1},\ldots, g_{l(n)}$, let W be the set of $g_j \in G_j$ such that $|COL_j| \leq 4(l(n) - n + 1)/\epsilon$, i.e. the set of functions which have zero or more, but not too many, collisions in COL_j.

The probability $Pr[A(x,h) \in COL_j]$ that A on input x and $h \in_R H_n$, returns a value in COL_j can be written as

$$\sum_{g \in W} Pr[A(x,h) \in COL_j \mid g_j = g]Pr[g] + Pr[A(x,h) \in COL_j \text{ and } g_j \notin W],$$

where $Pr[g]$ is the probability of choosing g from G_j under the uniform distribution. Thus, for $g \in W$ we have $Pr[g] = Pr[W]/|W|$, where $Pr[W] = \sum_{g \in W} Pr[g]$, and hence $Pr[A(x,h) \in COL_j]$ is equal to

$$\sum_{g \in W} Pr[A(x,h) \in COL_j \mid g_j = g]Pr[W]/|W| + Pr[A(x,h) \in COL_j \text{ and } g_j \notin W].$$

Let $g \in W$ and u be such that $g \circ g_{j+1} \circ \ldots \circ f_n(x) = g \circ g_{j+1} \circ \ldots \circ f_n(u)$ and $g_{j+1} \circ \ldots \circ f_n(x) \neq g_{j+1} \circ \ldots \circ f_n(u)$. Since g is a 2-1 function, the number of elements in COL_j is equal to the number of collisions the composition of the first $l(n) - (j+1) + 1$ universal$_2$ functions (that is in turn a universal$_2$ function), makes with u. The expected number of collisions for this composition, $\delta_{g_{j+1} \circ \ldots \circ g_{l(n)} \circ f_n}(u, \{y \neq u\})$, is equal to $(2^n - 1)/2^j$, which is less than 2. Thus, from the Markov's inequality it follows that given x if we randomly choose $g_{j+1} \in_R G_{j+1},\ldots, g_{l(n)} \in_R G_{l(n)}$, and g in G_j, then the probability, $1 - Pr[W] = Pr[|COL_j| > 4(l(n) - n + 1)/\epsilon]$ is less than $(1/2)\epsilon/(l(n) - n + 1)$. Since $Pr[A(x,h) \in COL_j \text{ and } g_j \notin W] \leq Pr[|COL_j| > 4(l(n) - n + 1)/\epsilon]$ and because $Pr[A(x,h) \in COL_j] \geq \epsilon/(l(n) - n + 1)$, it follows

$$\sum_{g \in W} Pr[A(x,h) \in COL_j \mid g_j = g]Pr[W]/|W| \geq \frac{1}{2}\frac{\epsilon}{l(n) - n + 1}$$

and thus

$$\sum_{g \in W} Pr[A(x,h) \in COL_j \mid g_j = g]\frac{1}{|W|} \geq \frac{1}{2}\frac{\epsilon}{l(n) - n + 1}\left(1 - \frac{1}{2}\frac{\epsilon}{l(n) - n + 1}\right). \quad (1)$$

Now, consider the algorithm A' that on input $z = f_n(w)$ where $w \in_R \{0,1\}^n$,

1. runs A to produce x (if $f_n(x) = z$ stop successfully –this is, of course, negligible);
2. for $i = n, ..., j-1, j+1, ..., l(n)$, randomly chooses $g_i \in_R G_i$;
3. randomly chooses $g_j \in_R G_j$ such that $g_j \circ g_{j+1} \circ ... \circ g_{l(n)-n+1} \circ f_n(x) = g_j \circ g_{j+1} \circ ... \circ g_{l(n)-n+1}(z)$;
4. gets y by running A on input $h = g_n \circ ... \circ g_{l(n)} \circ f_n$ and x;
5. outputs y.

Notice that the probability that $g_{j+1} \circ ... \circ g_{l(n)} \circ f_n(x) = g_{j+1} \circ ... \circ g_{l(n)}(z)$ is $1/2^j$, which is negligible; in this case we say that A' fails and we can stop it. Otherwise, with probability $p_s = 1 - 1/2^j$, $f^{-1}(z)$ belongs to COL_j (by the forced collision). Next we compute the probability of inversion of z when A' does not stop. For the rest of the calculation p_s will be a multiplicative factor of the successful event.

Denote by D the distribution on G_j, according to which g_j is chosen at step 3. This is not an uniform distribution, not even among the functions in G_j that have at least one collision $g_j \circ g_{j+1} \circ ... \circ g_{l(n)} \circ f_n(x) = g_j \circ g_{j+1} \circ ... \circ g_{l(n)} \circ f_n(y)$, $y \neq x$. Indeed, for any two functions g', g'' we have $D[g'] = (|COL_j(g')|/|COL_j(g'')|)D[g'']$, where $COL_j(g)$ is the set COL_j when $g_j = g$. That is the probability $D[g]$ of a function g to be chosen at step 3 is proportional to the number of collisions there are in $COL_j(g)$, thus it is dependent on the previous choices of $g_{j+1}, ..., g_{l(n)}$.

The number of elements in COL_j is given by $\delta_{g_{j+1} \circ ... \circ g_{l(n)} \circ f_n}(f^{-1}(z), \{y \neq f^{-1}(z)\})$, and its expected value when h has been chosen according to A''s algorithm, is equal to $(2^n - 1)/2^j$, which is less than 2. Thus, from Markov's inequality it follows that $D[W]$ is greater than or equal to $p_s\{1-(1/2)\epsilon/(l(n)-n+1)\}$. Let g' and g'' be two functions in W. From $D[g'] = (|COL_j(g')|/|COL_j(g'')|)D[g'']$, it follows that $D[g'] \geq p_s(1/|W|)(\epsilon/\{4(l(n)-n+1)\})\{1-(1/2)\epsilon/(l(n)-n+1)\}$.

The probability $Pr[A'(x,h) \in COL_j]$ that A' on input x and h chosen by A' in H_n, returns a value in COL_j is at least

$$\sum_{g \in W} Pr[A(x,h) \in COL_j \mid g_j = g]D[g],$$

which is greater than or equal to

$$\sum_{g \in W} Pr[A(x,h) \in COL_j \mid g_j = g]\frac{p_s}{4|W|}\left(\frac{\epsilon}{l(n)-n+1}\right)\left(1 - \frac{1}{2}\frac{\epsilon}{l(n)-n+1}\right).$$

Making use of (1), it follows that $Pr[A'(x,h) \in COL_j]$ is greater than

$$\frac{p_s}{8}\left(\frac{\epsilon}{l(n)-n+1}\right)^2\left(1 - \frac{1}{2}\frac{\epsilon}{l(n)-n+1}\right)^2.$$

The probability that $g_j \in W$ when it is chosen accordingly to A''s algorithm is $D[W] \geq p_s\{1-(1/2)(\epsilon/(l(n)-n+1))\}$. When A' at step 5 returns an element $y \in COL_j$ that collides with x, then the probability that $f(y) = z$ is $1/|COL_j|$

(since z is not an input to A). Hence the probability that $y = A'(x,h)$ satisfies $f(y) = z$ is at least

$$\frac{p_s^2}{32}\left(\frac{\epsilon}{l(n)-n+1}\right)^3\left(1-\frac{1}{2}\frac{\epsilon}{l(n)-n+1}\right)^3,$$

which is polynomially related to ϵ (notice that, say, $p_s > 1/4$). □

4 Further Reducing Complexity Assumptions

Next, we show how to construct UOWHF using one-way functions that are more general. First, we show that a function with a small preimage size gives us a UOWHF as well. This will be followed by a more general result: a function with the property that the expected size (when an element in the domain is randomly chosen) of the preimage of an element in the range is small gives us a UOWHF. We show also how to construct such a function if a regular function [GKL] is available, or even when a function where given an element in the range an estimate (with polynomial uncertainty) on the size of the preimage set is easily computable.

4.1 UOWHF Based on One-way Functions with Small Expected Preimage Size

Here we describe how to construct a UOWHF if a one-way function that has at most only polynomially many collisions on the average is available. We first define formally what we mean by small preimage size and then by small expected preimage size. Roughly speaking, the latter is a function f with the property that for a randomly chosen x, the expected size of the preimage of $f(x)$ is small. Then, we discuss why the previous scheme does not work with such functions. We describe a scheme for these functions and prove its correctness. In the next subsection we show how to construct such a function when it is only required that there is a feasible algorithm that when given an element z in the range, it gives a relatively good estimate on the size of the preimage set $f^{-1}(z)$.

The property of a small preimage size is shared by all 1–1 one-way functions, but also include for example, one-way functions based on the generalized factoring assumption of composite with two or more primes, such as modular squaring (whose inverse is extracting square roots) (see [GKL]).

Definition 6 *Let $r(\cdot)$ be a function from $\mathcal{N}^+$ to $\mathcal{N}^+$. A one-way function has a $r(n)$-preimage size if for each $x \in \{0,1\}^n$*

$$|f^{-1}(f(x))| \leq r(n).$$

Definition 7 *Let $r(\cdot)$ be a function from $\mathcal{N}^+$ to $\mathcal{N}^+$. A one-way function has an expected $r(n)$-preimage-size if when x is randomly chosen in $\{0,1\}^n$ the expected size of $f^{-1}(f(x))$ is at most $r(n)$.*

Definition 8 *A one-way function f has a small expected preimage-size if there is a polynomial p such that f has an expected $p(n)$-preimage-size.*

Let f be a one-way function with expected $p(n)$-preimage-size. Denote by $\delta_f(x, \{y \neq x\})$ the number of elements such that $f(y) = f(x)$, then Markov's inequality implies that $Pr[\delta_f(x, \{y \neq x\}) > t \cdot p(n)] < 1/t$, where the probability is over the choices of $x \in_R \{0,1\}^n$. The Markov's inequality essentially states that there is only a negligible probability that there are more than polynomially many collisions. This is an important property for the proof of our scheme for UOWHF.

Why the previous scheme does not work with small preimage size functions
Suppose we use the same scheme described earlier to construct UOWHF based on 1–1 functions, but we plug in a $p(n)$-preimage size function as the underlying one-way function. So, h is constructed as $g_n \circ g_{n+1} \circ ... \circ g_{l(n)} \circ f_n$, where each g_i is a hash function that shrinks the input by one bit. To prove its correctness we should derive a contradiction with the difficulty of inverting f; i.e. the ability to easily find a collision for the h would imply the ability to invert the f. It is immediate that this approach is doomed to failure. Indeed, suppose that there is a poly-time algorithm that on input x outputs y such that $f(x) = f(y)$ and $y \neq x$ (this is not in contradiction with the difficulty of inverting f). Then, this latter algorithm can be used to find a collision for h. Squaring modulo a composite is such a function f, so is any one-way function which is independent of part of its input and just applies to the rest of the argument. .

A provably secure scheme
We just saw the problem in dealing with functions that are not 1–1, as the difficulty of inverting does not rule out the possibility of easily finding collisions for the one-way function and may thus jeopardize the security of the h function that is based on it. Here we show how to deal with this problem.

Let $\delta > 0$ be a constant. Let $f = \{f_n\colon \{0,1\}^n \rightarrow \{0,1\}^{l(n)} \mid n \in N^+\}$ be a one-way function with the output length $l(n)$ and expected $p(n)$-preimage-size, where $p(\cdot)$ is a polynomial. Recall that $G_{n,\lfloor(\log n)^{1+\delta}\rfloor}$ and G_i are families of universal$_2$ functions from $\{0,1\}^n$ to $\{0,1\}^{\lfloor(\log n)^{1+\delta}\rfloor}$ and from $\{0,1\}^i$ to $\{0,1\}^{i-1}$, respectively. For a positive integer k, let $\widehat{H}_k$ be the set of functions $\hat{h}_k\colon \{0,1\}^k \rightarrow \{0,1\}^{k-1}$ defined as

$$\hat{h}_k(w) = g_k \circ g_{k+1} \circ ... \circ g_{l(k)} \circ f_k(w)$$

where $g_i \in G_i$, $i = k, ..., l(k)$. Let H_n be the set of functions $h\colon \{0,1\}^n \rightarrow \{0,1\}^{n-1}$ defined as

$$h(x) = g(x) \diamond (\hat{h}_{n-\lfloor(\log n)^{1+\delta}\rfloor} \circ ... \circ \hat{h}_n \circ h'(x))$$

where $g \in G_{n,\lfloor(\log n)^{1+\delta}\rfloor}$, $\hat{h}_i \in \widehat{H}_i$, $i = n - \lfloor(\log n)^{1+\delta}\rfloor, ..., n$, and $h' \in G_{n,n}$ is a universal$_2$ permutation.

To randomly choose an element in H_n, we first randomly select $g \in_R G_{n,\lfloor(\log n)^{1+\delta}\rfloor}$, then $h' \in_R G_{n,n}$, and finally $\hat{h}_i \in_R \widehat{H}_i$, $i = n - \lfloor(\log n)^{1+\delta}\rfloor, ..., n$, uniformly and independently from each other.

In the above scheme to compress c bits one needs to apply one-way functions only $\lfloor(\log n)^{1+\delta}\rfloor + c$ times.

Theorem 3 *$U = \bigcup_n H_n$, based on small expected preimage functions as above, is a UOWHF family.*

Proof. Suppose there is an algorithm A, that on input n produces an $x \in \{0,1\}^n$ and given a randomly chosen $h \in_R H_n$ outputs a y such that $h(x) = h(y)$ and $x \neq y$, with probability greater than ϵ. As for the previous proof of Theorem 2, we will describe a probabilistic poly-time algorithm A' that on input $z = f(w)$, where w has been randomly chosen, finds a u such that $f(u) = f(w)$, with probability polynomially related to ϵ.

It is unlikely that A on input x and $h \in_R H_n$ returns a value $y = A(x,h)$ such that $y \neq x$, $g(y) = g(x)$ and $\widehat{h}_n(y) = \widehat{h}_n(x)$. Indeed, when $\widehat{h}_n \in_R \widehat{H}_n$, $h' \in_R G_{n,n}$, and $g \in_R G_{n,\lfloor(\log n)^{1+\delta}\rfloor}$, the probability that there exists a y such that $f_n(h'(x)) \neq f_n(h'(y))$, $g(y) = g(x)$ and $\widehat{h}_n(y) = \widehat{h}_n(x)$ is at most $2^n/2^{n-1+\lfloor(\log n)^{1+\delta}\rfloor}$, which is negligible. Moreover, when $h' \in_R G_{n,n}$, and $g \in_R G_{n,\lfloor(\log n)^{1+\delta}\rfloor}$, the probability that there exists a y such that $f_n(h'(x)) = f_n(h'(y))$ and $g(y) = g(x)$, is at most $p(n)/2^{\lfloor(\log n)^{1+\delta}\rfloor}$ (by Markov's inequality) which is negligible. (This is indeed the reason why, in the definition, $\lfloor(\log n)^{1+\delta}\rfloor$ of the output bits of $h(x)$ have been chosen to be an hashed value of x through the universal$_2$ g.) Thus, we assume this is not the case (we can assume that for all sufficiently large n, this will happen with probability at least $1/2$, to be generous).

Given x and h, denote by COL_i, $n - \lfloor(\log n)^{1+\delta}\rfloor \leq i < n$, the set of y such that $g(x) = g(y)$, $\widehat{h}_i \circ ... \circ \widehat{h}_n(x) = \widehat{h}_i \circ ... \circ \widehat{h}_n(y)$ and $\widehat{h}_{i+1} \circ ... \circ \widehat{h}_n(x) \neq \widehat{h}_{i+1} \circ ... \circ \widehat{h}_n(y)$. Fix an integer j, $n - \lfloor(\log n)^{1+\delta}\rfloor \leq j < n$, such that with probability at least $(1/2)\epsilon/(\lfloor(\log n)^{1+\delta}\rfloor + 1)$ the algorithm A, on input x and $h \in_R H_n$, returns a value $A(x,h) \in COL_j$. Such a j must exist by the pigeonhole principle (wlog we assume now that j is known).

Let $\widehat{h}_j$ be the composition of $g_j \circ g_{j+1} \circ ... \circ g_{l(j)} \circ f_j$, and let $s = \widehat{h}_{j+1} \circ ... \circ \widehat{h}_n \circ f_n(x)$. Denote by C_i, $i = j, ..., l(j)$, the set of $y \in COL_j$, such that $g_i \circ g_{i+1} \circ ... \circ g_{l(j)} \circ f_j(s) = g_i \circ g_{i+1} \circ ... \circ g_{l(j)} \circ f_j(y)$ and, for $i < l(j)$, $g_{i+1} \circ ... \circ g_{l(j)} \circ f_j(s) \neq g_{i+1} \circ ... \circ g_{l(j)} \circ f_j(y)$. And denote by $C_{l(j)+1}$ the set of $y \in COL_j$ such that $f_j(s) = f_j(y)$. Fix an integer k, $j \leq k \leq l(j)+1$, such that with probability at least $(1/2)\epsilon/((\lfloor(\log n)^{1+\delta}\rfloor + 1)(l(j) - j + 2))$ the algorithm A, on input x and $h \in_R H_n$, returns a value $A(x,h) \in C_k$. Such a k must exist by the pigeonhole principle (wlog we assume now that k is known). We distinguish two cases: $k \leq l(j)$ and $k = l(j)+1$.

Suppose $k \leq l(j)$. Consider the algorithm A' that on input $z = f_j(w)$, where $w \in_R \{0,1\}^j$:

1. runs A to produce x;
2. randomly chooses $g \in_R G_{n,\lfloor(\log n)^{1+\delta}\rfloor}$;
3. for $i = n - \lfloor(\log n)^{1+\delta}\rfloor, ..., j-1, j+1, ..., n$, randomly chooses $\widehat{h}_i \in_R \widehat{H}_i$;
4. for $i = j, ..., k-1, k+1, ..., l(j)$, randomly chooses $g_i \in_R G_i$;

5. computes $s = \widehat{h}_{j+1} \circ ... \circ \widehat{h}_n \circ f_n(x)$, and randomly chooses $g_k \in_R G_k$ such that $g_k \circ g_{k+1} \circ ... \circ g_{l(j)} \circ f_j(s) = g_k \circ g_{k+1} \circ ... \circ g_{l(j)}(z)$;
6. constructs $\widehat{h}_j = g_j \circ g_{j+1} \circ ... \circ g_k \circ ... \circ g_{l(j)} \circ f_j$;
7. constructs h as the pair g and the composition $\widehat{h}_{n-\lfloor(\log n)^{1+\delta}\rfloor} \circ ... \circ \widehat{h}_n$
8. gets u by running A on input x and h;
9. outputs $y = \widehat{h}_{j+1} \circ ... \circ \widehat{h}_n(u)$.

Notice that there is a negligible probability that $f_j(s) = z$, in which case we invert z. Thus, assume this is not the case.

Every hash function (but g_k at step 5) is randomly chosen by A'. Let us denote by D the distribution under which such g_k is chosen by A'. If D were the uniform distribution and were independent from the choices of the other hash functions, then the probability of A' returning a colliding u in the set C_k would be the same as for A. The expected number of f values whose inputs are from $\{0,1\}^j$ that collide under the universal$_2$ hash function $g_k \circ ... \circ g_{l(j)}$ is at most $2^j/2^{k-1} \leq 2$; and thus it is unlikely that there are more than polynomially many collisions (from the Markov's inequality). Thus, when A gives a collision $u \in C_k$, the value $f \circ \widehat{h}_{j+1} \circ ... \circ \widehat{h}_n(u)$ is one of the f colliding values, and there is a non-negligible probability that this is exactly $z = f(w)$ (in which case we invert z). Unfortunately, D is not the uniform distribution but, as in the A''s algorithm of Theorem 2, it is close enough (for our purposes) to it. The proof of this case is essentially the same of Theorem 2 and, thus, is omitted here.

Now, suppose $k = l(j) + 1$. Here, the hypothesis is that A on input x and $h \in_R H_n$ returns, with probability at least $(1/2)\epsilon/((\lfloor(\log n)^{1+\delta}\rfloor + 1)(l(j) - j + 2))$, a $y \neq x$ such that $f_j \circ \widehat{h}_{j+1} \circ ... \circ \widehat{h}_n(x) = f_j \circ \widehat{h}_{j+1} \circ ... \circ \widehat{h}_n(y)$ and $\widehat{h}_{j+1} \circ ... \circ \widehat{h}_n(x) \neq \widehat{h}_{j+1} \circ ... \circ \widehat{h}_n(y)$. Now, we construct an algorithm A' that, by first running A, computes a pair of colliding and different elements in C_k. And then, by running algorithm A again, exploits the computed collision to try to invert an f value. More formally, the algorithm A' on input $z = f_{j+1}(w)$, where $w \in_R \{0,1\}^{j+1}$:

1. runs A to produce an initial x;
2. "computes a colliding pair $p_1 \neq p_2$ such that $f_j(p_1) = f_j(p_2)$"
   (a) randomly chooses $g \in_R G_{n,\lfloor(\log n)^{1+\delta}\rfloor}$;
   (b) for $i = n - \lfloor(\log n)^{1+\delta}\rfloor, ..., n$, randomly chooses $\widehat{h}_i \in_R \widehat{H}_i$;
   (c) constructs h as the pair g and the composition $\widehat{h}_{n-\lfloor(\log n)^{1+\delta}\rfloor} \circ ... \circ \widehat{h}_n$;
   (d) gets u by running A on input x and h;
   (e) computes the pair p_1, p_2 as $p_1 = \widehat{h}_{j+1} \circ ... \circ \widehat{h}_n(x)$ and $p_2 = \widehat{h}_{j+1} \circ ... \circ \widehat{h}_n(u)$;
3. for $i = j+2, ..., l(j+1)$, randomly chooses $g_i \in_R G_i$;
4. randomly chooses $g_{j+1} \in_R G_{j+1}$ such that $g_{j+1} \circ ... \circ g_{l(j+1)} \circ f_{j+1} \circ \widehat{h}_{j+2} \circ ... \circ \widehat{h}_n(x) = p_1$ and $g_{j+1} \circ ... \circ g_{l(j+1)}(z) = p_2$;
5. constructs $\widehat{h}_{j+1} = g_{j+1} \circ g_{j+2} \circ ... \circ g_{l(j+1)} \circ f_{j+1}$;
6. constructs h as the pair g and the composition $\widehat{h}_{n-\lfloor(\log n)^{1+\delta}\rfloor} \circ ... \circ \widehat{h}_n$;

7. gets v by running A on input x and h;
8. outputs $y = \widehat{h}_{j+2} \circ ... \circ \widehat{h}_n(v)$.

The probability that the pair p_1, p_2, computed at step 2, is such that $p_1 \neq p_2$ and $f_j(p_1) = f_j(p_1)$ is at least $(1/2)\epsilon/((\lfloor(\log n)^{1+\delta}\rfloor + 1)(l(j) - j + 2))$ (this is a bound on the probability of A outputting an element in C_k). Assume such a pair is successfully obtained at step 2. Let us denote by D the distribution under which g_k is chosen by A' at step 4. All other hash functions (but g_k) are randomly chosen by A'. As for the previous case, if D were the uniform distribution and were independent from the choices of the other hash functions, then the probability of A' returning a colliding v in the set C_k would be the same as for A. The expected number of f_{j+1} values that collide under the universal$_2$ hash function $g_{j+1} \circ ... \circ g_{l(j+1)}$ is at most $2^{j+1}/2^j = 2$, and thus it is unlikely that there are more than polynomially many collisions (from the Markov's inequality). Thus, at step 7 when A returns a collision $v \in C_k$, the value $f_{j+1} \circ \widehat{h}_{j+2} \circ ... \circ \widehat{h}_n(v)$ is one of the f_{j+1} colliding values, and there is a non-negligible probability that this is exactly $z = f_{j+1}(w)$ (in which case we invert z). Unfortunately, D is not the uniform distribution but, as in the A''s algorithm of Theorem 2 and in the previous case, it is close enough (for our purposes) to it. The formal proof of this case, based on similar techniques as in the previous proof, will be given in the final paper. □

4.2 One-way functions with almost-known preimage-size

In this subsection we show how to construct a small expected preimage-size function if there is a function which has a feasible algorithm that when given an element z in the range, gives a good estimate on the size of the preimage set $f^{-1}(z)$.

Definition 9 *A one-way function has a almost-known preimage-size if there is a polynomial p and a poly-time deterministic algorithm PRE_SIZE such that, on input $z = f(x)$, returns a value*

$$|f^{-1}(f(x))| - p(n) \leq PRE\_SIZE(z) \leq |f^{-1}(f(x))| + p(n)$$

for all $x \in \{0,1\}^n$, except a negligible fraction of them.

A particular case of almost-known preimage-size one-way function is a regular function [GKL]. This is a function where every image of an n-bit input has the same number of preimages of length n, another such function is decoding random linear codes (see [GKL]). Subset sum [IN] is another example of this function. Also a $p(n)$-preimage size function is a particular case of a almost-known preimage-size function. In [IN] the function subset-sum is used directly as a UOWHF, exploiting the instances of subset-sum which compress their argument (from n bits to $l(n) = (1 - \epsilon)n$). On the other hand, the most secure instance of subset-sum is shown to be length-preserving instances (from n bits to $l(n) = n$ bits or $l(n) = n + O(\log n)$); all subset-sum instances are "almost-known preimage-size" and (if one-way) can be used in our scheme.

Let f be a almost-known preimage-size one-way function, and PRE_SIZE be an algorithm that gives an approximation within $p(n)$ to the preimage size of f. Define $i(x) = \lceil \log(PRE\_SIZE(f(x)) + p(|x|)) \rceil$, and define f' as

$$f'(x \diamond g) = f(x) \diamond g \diamond [g(x)]_{i(x)}$$

where $x \in \{0,1\}^n$, $g \in G_{n,n}$ and $[y]_k$ is the first k bits of y.

Along the line of Lemma 5.1 of Impagliazzo, Levin, and Luby [ILL], it is possible to prove the following lemma.

Lemma 3 *f' is a one-way function, when $x \in_R \{0,1\}^n$ and $g \in_R G_{n,n}$. Moreover, given randomly chosen $x \in_R \{0,1\}^n$ and $g \in_R G_{n,n}$, the expected number of y, $y \neq x$, that collide $f'(y) = f'(x)$, is $|f^{-1}(f(x))|/2^{i(x)} \leq 1$.*

The range of f' may contain elements of different lengths. But, as mentioned in Section 2, it is an easy task to construct an equivalent function with the range of the same length. Thus, as a corollary of Theorem 3 we have the following general result.

Theorem 4 *If there is a almost-known preimage-size (or a regular) function then there is a UOWHF, and thus a signature scheme.*

5 On Various Notions of Security of Cryptographic Hash

In this section we give a number of definitions of hardness of one-way hash functions or one-way functions, with respect to collision finding. Our motivation is to demonstrate an equivalence among a large set of possible definitions so that any primitive which satisfies one of the conditions will automatically satisfy the other definitions. This will demonstrate the robustness of the definition of CFHF, and may be suggesting a possible way to attack the problem of finding new and less restrictive implementations of CFHF.

Next we define a few classes of functions according to the hardness properties which they satisfy.

We identify the following families of functions; essentially the idea is to classify all possible ways a collision is generated by a family of cryptographic hash functions. We assume that all the functions below are accessible and computable in polynomial-time.

- $\mathcal{H}$ is a Collision-free hash family.
- $\mathcal{F} = \{\mathcal{F}_k\}$ is a collection of pairs of functions such that there is a polynomial p and when $(f_1, f_2) \in_R \mathcal{F}_k$ $Pr[|\{(x,y) : f_1(x) = f_2(y)\}| > 0] > 1/q(k)$. But, for all polynomials q, for all efficient algorithms A, and for all sufficiently large k, $Pr[A(f_1, f_2) = (x,y) : f_1(x) = f_2(y)] < 1/q(k)$ when $f_1, f_2 \in_R \mathcal{F}_k$.
- $\mathcal{G} = \{\mathcal{G}_k\}$ is a collection of pairs of functions such that there is a polynomial q and when $(g_1, g_2) \in_R \mathcal{F}_k$ $Pr[|\{x : g_1(x) = g_2(x)\}| > 0] > 1/q(k)$.

But, for all polynomials p, for all efficient algorithms A, and for all sufficiently large k, $Pr[A(g_1, g_2) = x : g_1(x) = g_2(x)] < 1/p(k)$ when $g_1, g_2 \in_R \mathcal{F}_k$.

- $\mathcal{R}$ is a collection of functions such that there is a polynomial q and when $r \in_R \mathcal{R}$ $Pr[|\{x : r(x) = 0\}| > 0] > 1/q(|x|)$. But, for all polynomials p, for all efficient algorithms A, and for all sufficiently large k, $Pr[A(r) = x : r(x) = 0] < 1/p(k)$ when $r \in_R \mathcal{R}_k$.
- Given a fixed polynomial-time computable function g', $\mathcal{S}$ is a collection of functions such that there is a polynomial q and when $s \in_R \mathcal{S}$ $Pr[|\{x : s(x) = g'(x)\}| > 0] > 1/q(|x|)$. But, for all polynomials p, for all efficient algorithms A, and for all sufficiently large k, $Pr[A(s, g') = x : s(x) = g'(x)] < 1/p(k)$ when $s \in_R \mathcal{S}_k$.

The above functions demonstrate various ways of defining collisions of hash functions, assuming the above functions actually compress the size of their argument. As mentioned is section 2, we can assume that for a given function, range elements with preimages of the same length n, have the same length $l(n)$.

$\mathcal{H}$ is the original CFHF family, which implies that on the same function finding a collision between two different arguments is hard. $\mathcal{F}$ is a family in which a collision between two functions on two different arguments is hard to find, while $\mathcal{G}$ is the family in which for two different functions a collision on the same argument is hard to find. $\mathcal{R}$ is a family in which non-negligible fraction belongs to the kernel and it is hard to find an element in the preimage of the kernel (similarly it can defined with respect to any constant in range, the family $\mathcal{S}$ captures the hardness of finding an argument colliding with a given fixed efficiently computable function. Below we show that collision freeness is robust with respect to the exact notion of collision.

Theorem 5 *The following relations on the function families exist:*

- *The families $\mathcal{F}, \mathcal{G}, \mathcal{R}, \mathcal{S}$ are information theoretically equivalent.*
- *The families $\mathcal{F}, \mathcal{G}, \mathcal{R}, \mathcal{S}$ are information theoretically reducible to $\mathcal{H}$.*

Proof. The following reductions can be observed. $\mathcal{F} \Rightarrow \mathcal{R}$, randomly draw a pair (f_1, f_2) and set the function $r(x \diamond y) = f_1(x) \oplus f_2(y)$. $\mathcal{G} \Rightarrow \mathcal{R}$, randomly draw a pair (g_1, g_2) and set the function $r(x) = g_1(x) \oplus g_2(x)$. In both reductions, the probability of the kernel is polynomially related to the probability of the non-emptiness of the collision set. $\mathcal{R} \Rightarrow \mathcal{S}$, simply draw $r \in_R \mathcal{R}$ and set $s = r \oplus g'$. Similarly, $\mathcal{S} \Rightarrow \mathcal{R}$, simply draw $s \in_R \mathcal{S}$ and set $r = s \oplus g'$. $\mathcal{R} \Rightarrow \mathcal{F}$, draw a random r and set $f_1 = r$ and $f_2 = 0$, and the same reduction $\mathcal{R} \Rightarrow \mathcal{G}$. The probability of the non-emptiness of the kernel in the above reductions is polynomially related as well to the probability of the respected collision set. This concludes the proof of equivalence of $\mathcal{F}, \mathcal{G}, \mathcal{R}$, and $\mathcal{S}$.

Next we show that $\mathcal{H} \Rightarrow \mathcal{R}$. Let $\mathcal{H}_k = \{h : \{0,1\}^k \rightarrow \{0,1\}^{l(k)}\}$. Let $neq(x, y)$, where x, y are both k-bit long strings (otherwise, the function is undefined), a function which gives $1^{l(k)}$ if $x = y$ and $0^{l(k)}$ otherwise. Given $h \in \mathcal{H}_k$, define $r(x \diamond y) = h(x) \oplus h(y) \oplus neq(x, y)$. Notice that the probability

that the kernel of r is not empty is polynomially related to the probability of collision in h.

□

The above theorem justifies the generality of the definition of collision-free functions as the hardest condition among possible function families.

References

[BeM] Bellare M. and S. Micali, *How to Sign Given any Trapdoor Function*, Proceedings of the 20th Annual Symposium on the Theory of Computing, Chicago, Il, 1988, pp. 32-42.

[CW] J. L. Carter and M. N. Wegman, *Universal Classes of Hash Functions*, Journal of Computer and System Sciences 18 (1979), pp. 143-154.

[D] I. B. Damgård, *Collision Free Hash Functions and Public Key Signature Schemes*, Eurocrypt 1987.

[DH] W. Diffie and Hellman, *New Directions in Cryptography*, IEEE Trans. on Information Theory, vol. IT-22, 6 (1976), pp. 644-654.

[E] P. Elias, *Universal Codeword Sets and Representations of the Integers*, IEEE Trans. on Inform. Theory, vol. 21, n. 2, March 1975, pp. 194–203.

[Gi] M. Girault, *Hash-functions using modulo-N Operations*, Eurocrypt, 1987.

[Go] O. Goldreich, *Two Remarks Concerning the GMR Signature Scheme*, Crypto 1986.

[GKL] O. Goldreich, H. Krawczyk, and M. Luby, *On the existence of Pseudo-random Generators*, Proceedings of the 29th Symposium on the Foundation of Computer Science, 1988, pp. 12-24.

[GMRi] S. Goldwasser, S. Micali, and R. Rivest, *A secure digital signature scheme*, Siam Journal on Computing, Vol. 17, 2 (1988), pp. 281-308.

[GMY] S. Goldwasser, S. Micali, and A. C. Yao, *Strong signature schemes*, Proceedings of the 15th Annual Symposium on the Theory of Computing, Boston, MA, 1983, pp. 431-439.

[ILL] R. Impagliazzo, L. Levin, and M. Luby, *Pseudo-Random Generation from One-way Functions*, Proceedings of 21st STOC, May 1989.

[IL] R. Impagliazzo and M. Luby, *One-way Functions are Essential for Complexity Based Cryptography*, Proceedings of the 30th Symposium on the Foundation of Computer Science, 1989.

[IN] R. Impagliazzo and M. Naor, *Efficient Cryptographic Schemes Provably secure as Subset Sum*, Proceedings of the 30th Symposium on the Foundation of Computer Science, 1989.

[La] L. Lamport, *Constructing digital signatures from one-way functions*, SRI intl. CSL-98, October 1979.

[M] R. Merkle, *A Digital Signature based on Conventional Encryption Function*, Crypto 1987, Springer Verlag.

[M1] R. Merkle, *Secrecy, Authentication and Public Key Systems*, Ph.D. Thesis (1982), UMI Research Press, Ann Arbor, Michigan.

[M2] R. Merkle, *One-way Hash Functions and DES*, Crypto 1989.

[MH] R. Merkle and M. Hellman, *Hiding Information and Signature in Trapdoor Knapsack*, IEEE Trans. on Inform. Theory, vol. 24, n. 5, 1978, pp. 525–530.

[NY] M. Naor and M. Yung, *Universal One-way Hash Functions and their Cryptographic Applications*, Proceedings of 21st STOC, May 1989.

[R] M. O. Rabin, *Digital Signatures and Public Key Functions as Intractable as Factoring*, Technical Memo TM-212, Lab. for Computer Science, MIT, 1979.

[Ro] J. Rompel, *One-way Functions are Necessary and Sufficient for Signature*, STOC 90.

[RSA] R. Rivest, A. Shamir, and L. Adleman, *A Method for Obtaining Digital Signature and Public Key Cryptosystems*, Comm. of ACM, 21 (1978), pp. 120-126.

[Y] A. C. Yao, *Theory and Applications of Trapdoor functions*, Proceedings of the 23th Symposium on the Foundation of Computer Science, 1982, pp. 80-91.