A Construction for One Way Hash Functions and Pseudorandom Bit Generators

Babak Sadeghiyan
and
Josef Pieprzyk[1]

Department of Computer Science,
University College,
University of New South Wales,
Australian Defence Force Academy,
Canberra, A.C.T. 2600,
Australia.

Abstract

We prove that if f is a n-bit one-way permutation, i.e., it has some hard bits, a one-way permutation with $n-k$ provably simultaneous hard bits can be constructed with it. We apply this construction to improve the efficiency of Blum-Micali pseudo-random bit generator. Then, we apply the construction to propose a new approach for building universal one-way hash functions. This approach merges Damgard's design principle (or Merkle's meta-method) and the method proposed by Zheng, Matsumoto and Imai for the construction of hash functions for long messages.

1 Introduction

Consider a situation when a communicant A, sends a message to a receiver B. In order to protect the message against forgery, A should sign the message. If the length of the message is long, the efficient method is *transforming all the message to a short string with a public hash function h, then signing the hash value.*

The hash function simply maps messages of arbitrary length to some small fixed length, and should satisfy four properties:

1. h should have the property that for any given message x and hash value $h(x)$, finding another message y such that $h(y) = h(x)$, $x \neq y$, is computationally hard. In other words, a hash function has to be collision free.

[1]Support for this project was provided in part by Telecom Australia under the contract number 7027 and by the Australian Research Council under the reference number A48830241.

D.W. Davies (Ed.): Advances in Cryptology - EUROCRYPT '91, LNCS 547, pp. 431-445, 1991.

2. h should be one-way so that messages are not disclosed by their signatures.(Properties 1 and 2 are related to each other.)

3. h should have the property to be computed over the entire message (with different lengths).

4. h should destroy the homomorphic structure of the underlying public key cryptosystem used for the signature scheme (for a detailed discussion see [9],[14],[15],[21]).

Several approaches for constructing hash functions based on DES have been proposed (see [1]). Unfortunately, DES suffers from small key space and also has some undesired properties such as complementation property, i.e., $e_k(\overline{m}) = \overline{e_k(m)}$. In [20], it is shown that some of these hash functions are not collision free. Several other proposals based on DES appear in [19],[23],[18], [28],[29], which try to overcome these drawbacks. Some other hash functions based on RSA and squaring modulo n appear in [16],[10]. Attacks to some of these constructions appear in [6],[10]. Later, Damgard [7] constructed hash functions based on the existence of a claw free pair of permutations, which was the first for which collision freeness could be proven.

The current trend in cryptography is to provide the construction of basic primitives with general cryptographic assumptions that are also as weak as possible (it is theoretically important to base cryptographic primitives and basic tools on reduced complexity assumptions). It is also practically important to give efficient implementation of such constructions [24].

Two formal complexity-theoretic definitions have been suggested for cryptographic hash function families. The first family of hash functions, defined by Damgard, is the Collision Free Hash Functions (CFHF) or Collision Intractable Hash Functions (CIH). In this family, the function is given but finding a pair which map to the same output is difficult (see [8] for the precise definition). The second family, defined by Naor and Yung [22], is the Universal One Way Hash Functions (UOWHF). This family is weaker and if given an input and any function from the family, it is difficult to find another input which collides with it.

Naor and Yung [22] showed that a secure signature scheme reduces to the existence of UOWHF. They constructed a family of UOWHF from any one-way permutation and a family of strongly universal$_2$ hash functions, which was introduced in [5],[27], with collision accessibility property. In their construction, the one-way permutation provides the one-wayness of the UOWHF, and the strongly universal$_2$ family of hash functions performs the mapping to the small length output. When a member is chosen randomly and uniformly from the family, the output is distributed randomly and uniformly over the output space.

Santias and Yung [25] constructed the family of UOWHF from any 1 to 1 one-way function. They also proposed another construction from any one-way function with almost known pre-image size. Zheng, Matsumoto and Imai [32] built a family of UOWHF from any quasi-injection one-way function with the application of a pair-wise independent uniformiser and a family of strongly universal$_2$ hash functions. Rompel [24] gave a construction for one-way hash functions from any one-way function and proved that existence of one-way functions is a necessary and sufficient condition for constructing hash functions. However, although his work is theoretically optimal, it is less than practical.

In this area, each successive paper has assumed weaker conditions for the one-way function and has made hashing and additional procedures more complicated, such that Rompel's scheme can be constructed from any one-way function but needs too many additional procedures. In contrast, in this paper we construct a one-way permutation with some stronger properties but apply a simple hashing procedure, simply chopping or selecting arbitrarily some bits of the output.

On the other hand, for the construction of pseudo-random bit generators (PBG), Blum and Micali [3] discovered hard-core predicates b of functions f. Such $b(x)$ cannot be efficiently obtained,

given $f(x)$. They applied this notion to construct a PBG based on the intractability of the discrete logarithm problem. Yao [30] generalises this by showing that a PBG can be constructed from any one-way permutation. He transforms any one-way permutation into a more complicated one which has a hard-core predicate. Similarly, later works in this area have tried to make generalisations and assume weaker conditions (for example see [12], [13]).

In this paper, we present a method such that given an n-bit one-way permutation, i.e., it has some hard bits, a one-way permutation with n hard bits can be constructed, which we call a strong permutation. We apply this strong permutation to present a construction for pseudo-random bit generators with maximum efficiency, based on the Blum-Micali pseudo-random bit generator. We also present a method to build a universal one-way hash function from the strong permutation. Hence, given a one-way permutation, we can construct both an efficient pseudo-random generator and a universal one-way hash function. Zheng, Matsumoto and Imai [31] revealed a duality between pseudo-random bit generators and UOWHF. Applying the revealed duality, they presented a construction for UOWHF which is a dual of the construction of Blum-Micali PBG. We show that by the application of the strong permutation, Zheng et al.'s scheme and Damgard's design principle for construction of hash functions merge with each other, and would yield the same result. As the result, our proposal yields an algorithm that can be used both for generating pseudorandom bits, and hashing long messages. This has a practical significance, since it would not be necessary to use two different algorithms for implementing these two cryptographic tools.

2 Notations

The notation we use here is similar to [31]. The set of all integers is denoted by N. Let $\Sigma = \{0,1\}$ be the alphabet we consider. For $n \in N$, Σ^n is the set of all binary strings of length n. The concatenation of two binary strings x, y is denoted by $x \parallel y$. The length of a string x is denoted by $\mid x \mid$.

Let l be a monotone increasing function from N to N and f a function from D to R, where $D = \bigcup_n D_n$, $D_n \subseteq \Sigma^n$ and $R = \bigcup_n R_n$, $R_n \subseteq \Sigma^{l(n)}$. D is called the domain and R the range of f. Denote by f_n the restriction of f on Σ^n. f is a permutation if each f_n is a 1 to 1 and onto function. f is polynomial time computable if there is a polynomial time algorithm computing $f(x)$ for all $x \in D$. The composition of two functions f and g is defined as $f \circ g(x) = f(g(x))$. The i-fold composition of f is denoted by $f^{(i)}$.

A (probability) ensemble E, with length $l(n)$, is a family of probability distributions $\{E_n \mid E_n : \Sigma^{l(n)} \rightarrow [0,1], n \in N\}$. The uniform ensemble U with length $l(n)$ is the family of uniform probability distributions U_n, where each U_n is defined as $U_n(x) = \frac{1}{2^{l(n)}}$, for all $x \in \Sigma^{l(n)}$. By $x \in_E \Sigma^{l(n)}$ we mean that x is randomly selected from $\Sigma^{l(n)}$ according to E_n, and in particular by $x \in_r S$ we mean that x is chosen from the set S uniformly at random. E is samplable if there is an algorithm M that on input n, outputs an $x \in_E \Sigma^{l(n)}$, and polynomially samplable if the running time of M is also polynomially bounded.

3 Preliminaries

Definition 1 *A statistical test is a probabilistic algorithm T that on an input x, where x is an n-bit string, halts in $O(n^t)$ and outputs a bit $0/1$, where t is some fixed positive integer.*

Definition 2 *Let l be a polynomial, and E^1 and E^2 be ensembles both with length $l(n)$. E^1 and E^2 are called indistinguishable from each other, if for each statistical test T, for each polynomial Q, for all sufficiently large n,*

$$| Prob\{T(x_1) = 1\} - Prob\{T(x_2) = 1\} | < \frac{1}{Q(n)}$$

where $x_1 \in_{E^1} \Sigma^{l(n)}$, $x_2 \in_{E^2} \Sigma^{l(n)}$.

Definition 3 *A polynomially samplable ensemble E is pseudorandom if it is indistinguishable from the uniform ensemble U with the same length.*

Definition 4 *Let $f : D \rightarrow R$, where $D = \bigcup_n \Sigma^n$ and $R = \bigcup_n \Sigma^{l(n)}$, be a polynomial time computable function. We say that f is one-way if for each probabilistic polynomial time algorithm M, for each polynomial Q and for all sufficiently large n,*

$$Prob\{f_n(M(f_n(x))) = f_n(x)\} < \frac{1}{Q(n)}$$

where $x \in_U D_n$.

Note that the one-way property of a function is relative to a specific model of computation with a specific amount of computing resources.

Definition 5 *We say we have a computing resource for k bits if given the output of a one-way function and $n - k$ bits of the input string, one can define the remaining k bits of the input string by exhaustive search.*

For the remaining of this paper we assume that we have a computing resource for at most k bits.

4 Hard Bits

If a function f is one-way then given $f(x)$ the argument x must be unpredictable. If every bit of the argument x were easily computable from $f(x)$, then f would not be a one-way function. Therefore, some specific bits of the argument are unpredictable, and we cannot guess them better than by flipping a coin. We call these bits hard bits of f.

Definition 6 *Let $f : D \rightarrow R$ be a one-way function, where $R = \bigcup_n \Sigma^n$ and $D = \bigcup_n \Sigma^{l(n)}$. Let $i(n)$ be a function from N to N with $1 \leq i(n) \leq n$. If for each probabilistic polynomial time algorithm M, for each Q and for all sufficiently large n,*

$$Prob\{M(f_n(x)) = x'_{i(n)}\} < \frac{1}{2} + \frac{1}{Q(n)}$$

where $x \in_r \Sigma^n$ and $x'_{i(n)}$ is the $i(n)$-th bit of an $x' \in \Sigma^n$ satisfying $f(x) = f(x')$, then $i(n)$-th bit is a hard bit of f [91].

Note that the definition of hard bits implies that a hard bit depends on all bits of $f(x)$ under f^{-1}, where f^{-1} is a hard problem.

Lemma 1 *The number of hard bits defines the difficulty of inverting a one-way function.*

Proof : Assume that only a small number of bits of a function are hard bits and, when the output is given, we can obtain every remaining bit with a probability better than $\frac{1}{2} + \frac{1}{Q(n)}$ in polynomial time. A probabilistic algorithm M that first predicts the easy bits and then does an exhaustive search for finding hard bits can inverse the function f in polynomial time with a probability at least better than $\frac{1}{Q(n)}$. For example, consider that a function has been proven to have only $\log_2(n)$ hard bits and $n = 512$ then only 9 bits are hard. If we have a computing resource for more than 9 bits, which we usually have, then given the output, the input can be obtained in polynomial time with a probability better than $\frac{1}{Q(n)}$. □

Hence, a one-way function should have at least $k + 1$ hard bits.

Lemma 2 *All the hard bits are independent of one another.*

Proof : (By contradiction) assume that the i_1, i_2-th bits are hard bits that are dependent on each other and there is a probabilistic algorithm M that can calculate i_1-th bit given both $f(x)$ and i_2-th bit with a probability better than $\frac{1}{Q(n)}$. Then, we can construct a probabilistic algorithm M' for guessing i_1-th bit.

Algorithm M':

1. Guess i_2-th bit with flipping a coin (guess with probability 0.5).
2. Given i_2-th bit and $f(x)$, run M and find i_1-th bit.

then $Prob\{M'(f(x)) = x_{i_1}\} > \frac{1}{2} + \frac{1}{Q(n)}$; which is a contradiction. □

From the above Lemma we draw the following corollary.

Corollary 1 *Let $f : D \rightarrow R$ be a one-way function, where $D = \bigcup_n \Sigma^n$ and $R = \bigcup_n \Sigma^{l(n)}$. Assume f has t hard bits, $t < n - k$, and $j < k$ of them and $f(x)$ are given, we cannot predict any of the remaining $t - j$ hard bits with a probability better than $\frac{1}{2} + \frac{1}{Q(n)}$.*

Definition 7 *Let l be a polynomial, and E be an ensemble with length $l(n)$. We say that E passes the next bit test if for each statistical test T, for each polynomial Q, for all sufficiently large n, the probability that on input the first i bits of a sequence x randomly selected according to E and $i < l(n)$, T outputs the $(i+1)$th bit of x is:*

$$Prob\{T(x_1, \ldots, x_i) = 1\} < \frac{1}{2} + \frac{1}{Q(n)}$$

where $x \in_E \Sigma^{l(n)}$.

The following theorem is derived from [Yao 82] and has been stated in [2], [3], [11] in different terms.

Theorem 1 *Let E be an polynomially samplable ensemble, the following statements are equivalent:*
(i) E passes the next bit test.
(ii) E is indistinguishable from the uniform ensemble U.

In other words, the indistinguishability test is equivalent to the unpredictability test.

Corollary 2 *Assume that $f : D \to R$ is a one-way function, where $D = \bigcup_n \Sigma^n$ and $R = \bigcup_n \Sigma^{l(n)}$. Also assume that $i_1, i_2, \ldots, i_t$ are functions from N to N, with $1 \leq i_j(n) \leq n$ for each $1 \leq j \leq t$, $t < k$ and each i_j denotes a hard bit of f. Denote by E_n^1 and E_n^2 the probability distributions defined by the random variables $x_{i_t(n)} \ldots x_{i_2(n)}\, x_{i_1(n)} \parallel f(x)$ and $r_t \ldots r_2\, r_1 \parallel f(x)$ respectively, where $x \in_r \Sigma^n$, $x_{i_j(n)}$ is the $i_j(n)$-th bit of x and $r_j \in_r \Sigma$. Let $E^1 = \{E_n^1 \mid n \in N\}$ and $E^2 = \{E_n^2 \mid n \in N\}$, then E^1 and E^2 are indistinguishable from each other.*

Proof : From Corollary 1, it can be concluded that every string of $t < k$ hard bits passes the next bit test. This is equivalent to saying that given $f(x)$, any string of $t < k$ hard bits is indistinguishable from a string chosen uniformly at random from Σ^t, according to Theorem 1. □

In other words, given $f(x)$, any string of $t < k$ hard bits is indistinguishable from random strings. Such hard bits are called simultaneous hard bits of f. Note that the maximum number of simultaneous hard bits of any one way function can not be more than $n - k$.

Blum and Micali discovered the notion of hard core predicates of functions and applied it to construct pseudorandom bit generators (PBG).

Definition 8 *Let l be a polynomial with $l(n) > n$. A pseudorandom bit generator is a deterministic polynomial time function g that upon receiving a random n-bit input, extends it into a sequence of $l(n)$-bit pseudorandom bits $b_1, b_2, \ldots, b_{l(n)}$ as the output.*

In other words:

1. Each bit b_k is easy to compute.
2. The output bits are unpredictable, in other words the output string passes the next bit test, i.e., given the generator g and the first s output bits $b_1, \ldots, b_s$, but not the input string, it is computationally infeasible to predict the $(s+1)$th bit in the sequence [3].

The following theorem describes Blum-Micali PBG [3].

Theorem 2 *Let l be a polynomial with $l(n) > n$, and let f be a one-way permutation on $D = \bigcup_n \Sigma^n$ and $i(n)$-th bit is proven to be a hard bit of f. Let g_n be a function defined as follows:*

1. *Generate the sequence $f_n^{(1)}(x), f_n^{(2)}(x), \ldots, f_n^{(l(n))}(x)$, where $x \in \Sigma^n$.*
2. *From right to left (!), extract i-th bit from each element in the above sequence and output that bit.*

so, $g_n(x) = b_{l(n)}(x) \ldots b_2(x)\, b_1(x)$ where $x \in \Sigma^n$ and $b_j(x) =$ (the i-th bit of $f_n^{(j)}(x)$). The $g = \{g_n \mid n \in N\}$ is a pseudorandom bit generator extending n-bit into $l(n)$-bit output strings.

If $i_1(n), \ldots, i_t(n)$-th bits are simultaneous hard bits of f, then the efficiency of g can be improved by defining the $b_j(x)$ to be a function which extracts all known simultaneous hard bits of $f^{(j)}(x)$.

In [2], it has been proven that the $\log_2(n)$ least significant bits of RSA and Rabin encryption functions are simultaneously hard. Hence, if we use RSA or Rabin functions instead of the one-way permutation, with each iteration of the function we can extract $\log_2(n)$ bits. For example, if n is equal to 512 and we would like to produce a 512 bit pseudorandom string, we should iterate the one-way function for $\lceil \frac{l(n)}{\log_2(n)} \rceil = \lceil \frac{512}{\log_2(512)} \rceil = 57$ times. If a one-way permutation has more known hard bits, we can use it instead of RSA or Rabin function and obtain a better efficiency.

5 A Strong One Way Permutation

In this section we construct a one-way permutation with maximum number of hard bits, which can be used for the construction of both the Blum-Micali pseudorandom bit generator and one-way hash functions. Before describing the construction some preliminary definitions are given.

Definition 9 *A transformation is called complete if each output bit depends on all input bits. In other words, the simplest Boolean expression for each output bit contains all the input bits.*

Definition 10 *If the inverse of a complete transformation is also complete, it is described as being two way complete. In other words, each output bit depends on all the input bits and vice-versa.*

Lemma 3 *If a permutation is complete, then it is also two way complete.*

Definition 11 *If the correlation between two binary variables is zero, they are called independent variables.*

(See [26], for the definition of correlation.)

Definition 12 *Let v be a complete permutation and all the output bits be pairwise independent. We call v a perfect permutation.*

Kam and Davida [17] presented a method where an entire substitution-permutation network could be guaranteed to be complete if all the substitution boxes used in the procedure were complete. DES is an example of a complete cryptographic transformation. Since DES is reversible and the reverse function (decryption) has the same structure as encryption, DES is a two way complete transformation. Webster and Tavares [26] showed that there is very little correlation between output variables of DES. So, we can conclude that DES is an example of a perfect permutation, in our definitions. Brown [4] has used the known design criteria of DES to build an extended 128-bit DES and has shown that his scheme has similar cryptographic properties to DES. Extending the DES structure for more bits, for example 512 bits, has the disadvantage that the running time would be relatively high and would be comparable to public key cryptosystem. For the following theorems, we use a two way complete permutation such that only $k+1$ output bits are independent of other bits and we call it a $k+1$-bit perfect permutation, which has much looser requirements than a perfect permutation. If we consider $k = 63$, a $k+1$-bit perfect permutation can easily be constructed with the Kam and Davida method, plus a single DES block.

Lemma 4 *Let f be an n-bit one-way permutation and V be the set of all n-bit permutations, then $m = f \circ v \circ f$ is also a one-way permutation with a probability better than $1 - \frac{1}{Q(n)}$, when $v \in_r V$.*

Proof : Both f and v are permutations and f is a one-way permutation, so the result of their composition would be a permutation. The probability that m would not be a one way permutation is equal to the probability of finding f^{-1} (or any linear function of f^{-1}) from V by chance and is equal to $\frac{1}{Q(n)}$. □

If we put some conditions on v and f, m can be made to be a permutation with the desired properties.

Theorem 3 *Let $m : D \rightarrow D$ be a one-way permutation where $D = \bigcup_n \Sigma^n$ and $m = f \circ v \circ f$, where f is a one-way permutation, i.e., it has at least $k+1$ hard bits, and v is a $k+1$-bit perfect permutation where the positions of independent output bits comply with the position of hard bits of f. For each probabilistic polynomial time algorithm M, for each Q and for all sufficiently large n,*

$$Prob\{M(m(x)) = x_i\} < \frac{1}{2} + \frac{1}{Q(n)}$$

where $x \in_r \Sigma^n$ and x_i is the i-th bit of the x, and $1 \leq i \leq n$. In other words, each bit of x is a hard bit of m.

Proof : (By contradiction) we show that if an algorithm could find x_i, it would be able to invert f. For simplicity of notation, we indicate the first one-way function with f_1 and the second one with f_2, so $m = f_2 \circ v \circ f_1$. Assume that M is an algorithm that given $m(x)$, can predict x_i with a probability bigger than $\frac{1}{2} + \frac{1}{Q(n)}$ (x_i is not a hard bit of m). Two situations may arise;

(a) when x_i is not a hard bit of f_1:

Since the i-th bit is not a hard bit of f_1, then given $f_1(x)$, there exists an algorithm M' that can find the i-th bit with a probability bigger than $\frac{1}{2} + \frac{1}{Q(n)}$.

Without loss of generality, consider v to be an invertible permutation. Due to the two way completeness property of v, all bits of $v \circ f_1(x)$ depend on all bits of $f_1(x)$ and vice-versa. So, to obtain $f_1(x)$, we need to know all bits of $v \circ f_1(x)$. Since v is an invertible function in polynomial time, then given $v \circ f_1(x)$, it is possible to find the i-th bit of x,

$$Prob\{M'(v \circ f_1(x)) = x_i\} > \frac{1}{2} + \frac{1}{Q'(n)}$$

the probability equation simply says that we can predict x_i by tossing a coin with probability $1/2$ or estimating it given $v \circ f_1(x)$ with a probability better than $1/Q'(n)$. In other words,

$$Prob\{\text{estimating } x_i \mid v \circ f_1(x)\} > \frac{1}{Q'(n)}$$

Without loss of generality, we assume that f_2 is a one way permutation such that given a $f_2(y)$, we can guess $n-k-1$ bits of y efficiently. Moreover, the $k+1$ independent bits of v comply with hard bits of f_2, and knowing some other bits of $v \circ f_1(x)$ (i.e., other than independent output bits of v) and v, we cannot calculate all bits of $v \circ f_1(x)$. In accordance with the assumption that the i-th bit is not a hard bit of m, the following is also held:

$$\begin{aligned} \frac{1}{Q(n)} &< Prob\{\text{estimating } x_i \mid f_2 \circ v \circ f_1(x)\} \\ &= Prob\{\text{estimating } x_i \mid v \circ f_1(x)\}.Prob\{\text{obtaining } v \circ f_1(x) \mid f_2 \circ v \circ f_1(x)\} \end{aligned}$$

Since the multiplication of two polynomial expressions is another polynomial expression, then for some polynomial Q'', the following holds:

$$Prob\{\text{obtaining } v \circ f_1(x) \mid f_2 \circ v \circ f_1(x)\} > \frac{1}{Q''(n)}$$

This is equivalent to inverting f_2 and contradicts our assumption that f_2 is a one-way permutation.

(b) when the x_i is a hard bit of f_1:

by performing a procedure similar to the case *(a)*, it is obvious that the i-th bit should also be a hard bit of m. □

Lemma 5 *Let $m = f_2 \circ v \circ f_1$ be a one way permutation defined in Theorem 3. In addition, consider f_1 to be a one way permutation such that given a $t < n - k$ bits of x, no $\ell > k$ bits of $f(x)$ can be guessed with a probability better than $\frac{1}{2^k}$, then given $m(x)$ and the $t < n - k$ bits of x, $m(x)$ can not still be reversed.*

Proof : Since $t < n - k$ bits of x is known, the value of $f_1(x)$ can be guessed with a probability equal to $\frac{1}{2^{n-t}}$, where $n - t > k$. Hence, with v being a two way complete permutation, any bit of $v \circ f_1(x)$ can not be estimated with a probability better than $\frac{1}{2^{n-t}} < \frac{1}{2^k}$. On the other hand we assumed that given $f_2 \circ v \circ f_1(x)$, $n - k - 1$ bits of $v \circ f_1(x)$ could be guessed efficiently. Since the position of hard bits of f_2 comply with the positions of independent bits of v, given $n - k - 1$ bits of $v \circ f_1(x)$, we can not still estimate the $k + 1$ independent output bits of v with a probability better than $\frac{1}{2^{k+1}}$. Then the only possibility for reversing m is that the hard bits of f_2 and the $t < n - k$ bits of x be related to each other with some function such that revealing the t bits of x makes estimating the hard bits of f_2 probable. Such possibility has been excluded by assuming that f_1 is a one way permutation such that given a $t < n - k$ bits of x, no $\ell > k$ bits of $f(x)$ can be guessed with a probability better than $\frac{1}{2^k}$. Because, even if $v \circ f_1(x)$ and $f_1(x)$ are related to each other with a system of linear equations, knowing $n - k - 1$ bits of $v \circ f_1(x)$ and $\ell < k$ bits of the $f_1(x)$, the system of equations can not still be solved. □

Note that, the conditions of Lemma 5 for f_1 only excludes one way permutations that split into two or more parts, for example $f(x_1 \parallel x_2) = x_1 \parallel g(x_2)$.

Lemma 6 *Let $m = f_2 \circ v \circ f_1$ be the one way permutation defined in Theorem 3. In addition, consider f_1 to be a one way permutation such that given any string of $t < n - k$ bits of x, any $\ell < k$ bits of $f_1(x)$ can not be evaluated with a probability better than $\frac{1}{2^k}$, then given $m(x)$ and any string of $t < n - k$ bits of x, m can not be reversed.*

Lemma 6 simply suggested a construction for a one-way permutation m such that each bit of x is a hard bit of m and given any $t < n - k$ bits of x and $m(x)$, m can not be reversed. We call such a permutation m a *strong one-way permutation* or simply a *strong permutation*. The following corollary can be drawn from Lemma 6.

Corollary 3 *Assume that $m : D \to D$ is a strong one-way permutation, where $D = \bigcup_n \Sigma^n$. Also assume that $i_1, i_2, \ldots, i_t$ are functions from N to N, with $1 \leq i_j(n) \leq n$ for each $1 \leq j \leq t$, $t < n - k$. Denote by E_n^1 and E_n^2 the probability distributions defined by the random variables $x_{i_t(n)} \ldots x_{i_2(n)}\, x_{i_1(n)} \parallel m(x)$ and $r_t \ldots r_2\, r_1 \parallel m(x)$ respectively, where $x \in_r \Sigma^n$, $x_{i_j(n)}$ is the $i_j(n)$-th bit of x and $r_j \in_r \Sigma$. Let $E^1 = \{E_n^1 \mid n \in N\}$ and $E^2 = \{E_n^2 \mid n \in N\}$, then E^1 and E^2 are indistinguishable from each other.*

In other words, any string of $t < n - k$ bits of x is indistinguishable from random strings.

We can now construct an efficient Blum-Micali pseudo-random bit generator with the strong one-way permutation suggested in Theorem 3.

Theorem 4 *Let l be a polynomial with $l(n) > n$ and m be a strong one-way permutation. Let g be a function defined as follows:*

1. *Generate the sequence $m_n^{(1)}(x), m_n^{(2)}(x), \ldots, m_n^{(l(n))}(x)$, where $x \in \Sigma^n$.*
2. *From right to left, extract $n - k - 1$ bits from each element in the above sequence and output them.*

Then g is a pseudorandom bit generator extending n-bit into $(n-k-1)l(n)$ bit output strings.

Since we have a computing resource for k bits then the above scheme yields the maximum possible efficiency. If $k = 128$ (!) and n is 512, then with 2 iterations of m, or 4 iterations of f, we can extract 766 pseudorandom bits. This yields nearly 192 pseudo-random bits per iteration of f, which is 21 times more efficient than using RSA or Rabin function with the scheme described in Theorem 2.

Note that since the output string is pseudorandom, we can also draw the following corollary.

Corollary 4 *The $n-k-1$ extracted bits of each iteration is distributed uniformly and randomly in Σ^{n-k-1}.*

6 UOWHF Construction and PBG

Damgard in [8] suggested the use of pseudorandom bit generators for hash functions and extraction of a small portion of the output string, due to their one-way property, provided that the collision freeness of the concrete instance is analysed. Moreover, Zheng et al. [31] revealed a duality between the construction of pseudorandom bit generators and one-way hash functions. We show that the construction presented in Theorem 5 for PBG, can also be used for the construction of UOWHF. Before entering this discussion we give the background and some preliminary definitions.

There are two kinds of one-way hash functions, i.e., universal one way hash functions, or weak one-way hash functions, and collision free hash functions, or strong one-way hash functions. The main property of the former is that given a random initial string x, it is computationally infeasible to find a different string y that collides with x, i.e., $h(x) = h(y)$. The main property of the latter is that it is computationally difficult to find a pair (x, y) of strings, $x \neq y$, that collide with each other.

In universal one-way hash functions, there is no guarantee that it is computationally infeasible to find pairs of input that map onto the same output and for some inputs $z \neq z'$ might $h(z) = h(z')$. However, there should not be too many z, z' pairs. So, choosing x randomly should make it unlikely that anyone can find an x' such that $h(x) = h(x')$ [19]. However, if we assume that h is random, i.e., hashing is accomplished by looking up the correct value in a large table of random numbers, then it is possible to choose x in a non-random way since any method of choosing x that does not depend on h is random with respect to h.

Another problem with universal one-way hash functions is that repeated use weakens them. To deal with this problem, we can simply define a family of one-way hash functions with the property that each member h_i of the family is different from all other members, so any information about how to break h_i will provide no help in breaking h_j for $i \neq j$ (see [19]). If the system is designed so that every use of a weak one-way hash function is parameterized by a different parameter, then the overall system security can be kept high. Naor and Yung [22] constructed such a family of functions with a one-way permutation and a strongly universal$_2$ family of hash functions. The precise definition of UOWHF is given in Definition 13 below.

6.1 Preliminaries

Let l be a polynomial with $l(n) > n$, H is a family of hash functions defined by $H = \bigcup_n H_n$, where H_n is a set of functions from $\Sigma^{l(n)}$ to Σ^n. For two strings $x, y \in \Sigma^{l(n)}$ with $x \neq y$, we say that x

and y collide under $h \in H_n$ or (x, y) is a collision pair for h, if $h(x) = h(y)$. H is polynomial time computable if there is a polynomial time algorithm computing all $h \in H$, and accessible if there is a probabilistic polynomial time algorithm that on input $n \in N$ outputs uniformly at random a description of $h \in H_n$. Let F be a collision finder. F is a probabilistic polynomial time algorithm such that on input $x \in \Sigma^{l(n)}$ and $h \in H_n$ outputs either ? (cannot find) or a string $y \in \Sigma^{l(n)}$ such that $x \neq y$ and $h(x) = h(y)$.

Definition 13 *Let H be a computable and accessible hash function compressing $l(n)$-bit input into n-bit output strings and F a collision string finder. H is a universal one-way hash function if for each F, for each Q and for all sufficiently large n,*

$$Prob\{F(x, h) \neq ?\} < \frac{1}{Q(n)}$$

where $x \in \Sigma^{l(n)}$ and $h \in_r H_n$. The probability is computed over all $h \in_r H_n$, $x \in \Sigma^{l(n)}$ and the random choice of F.

6.2 UOWHF Based on the Strong One-Way Permutation

Theorem 5 *Assume that $m : D \rightarrow D$ is a strong one-way permutation, where $D = \bigcup_n \Sigma^n$, and $\text{chop}_1 : \Sigma^n \rightarrow \Sigma^{n-1}$ simply chops the last bit, then $h = \text{chop}_1 \circ m$ is a universal one-way hash function.*

Proof : (By contradiction), assume that there is a probabilistic algorithm F that can find a collision, then we show that we can make an algorithm that can invert m. Consider that we first choose an x at random, then run m on x to get $m(x)$, then we obtain $h(x) = \text{chop}_1(m(x))$. There is only one element that can collide with $m(x)$ under chop_1. This element differs with $m(x)$ in one bit. Let us notate this element $m(y)$. If a collision finder can find an y such that collides with x under h with probability bigger than $\frac{1}{Q(n)}$, it can obtain y from $m(y)$ with the same probability. This contradicts our assumption that m is a one-way permutation. □

Lemma 7 *If we define $\text{1chop} : \Sigma^n \rightarrow \Sigma^{n-1}$ simply to chop one bit and the position of chopped bit is defined in the description of the function and can be any bit, then $h = \text{1chop}(m(x))$ is also a universal one-way hash function.*

Proof Sketch: The problem of finding a collision for h, defined in Lemma 7, can be reformulated to finding x, y and $x \neq y$, such that $m(x)$ and $m(y)$ match at all bits except at the one defined in the definition of *1chop* function. By repeating a procedure similar to the procedure for the proof of Theorem 5 the claim of Lemma can be shown to be true. □

Since according to Corollary 3 and Corollary 4 the output of m is distributed uniformly and randomly in Σ^n, then for finding y with exhaustive search, we need to perform 2^{n-1} operations on the average. If this much computation is bigger than 2^k, then it is infeasible to find the collision.

If we chop t bits of $m(x)$, then there are $(2^t - 1)$ elements which collide with x under h. If these elements are distributed randomly in 2^n elements; then, we need to do 2^{n-t} search operations to find a collision for x. Since our computational resource can do at most 2^k search operations then t should be less than $n - k$.

Corollary 5 *Let* $\text{chop}_t : \Sigma^n \to \Sigma^{n-t}$ *simply to chop* t *last bits and* $t < n - k$*, then* $h = \text{chop}_t \circ m$ *is a universal one-way hash function.*

Note that the scheme described in the above corollary increases the efficiency of the hash function scheme, so for hashing long messages, we need to do less iterations. We can also generalise the above scheme by introducing *tchop* to be a function which chops t bits of the output. In this case, we need $(n - k - 1)\log_2 n$ bits to specify the positions of the chopped bits.

6.3 Parameterization

Since the hash function presented in Corollary 5 is a universal one-way hash function, we should parameterize it to make it secure in a practical scenario. The parameterization can be done in two different ways:

1. we can parameterize h by selecting v from a family of $k+1$-bit perfect permutations. Then $H = \{h = \text{chop}_t \circ f \circ v \circ f \mid v \in V_n\}$ where V_n is the $k+1$-bit perfect permutation family and chop_t simply chops the t last bits.

2. we can parameterize h by selecting the function for the compressing procedure from a family of hash functions. We may consider this family to be a family of *chop* functions. In this case, the number of bits required to specify a member of the family is at most equal to $(n - k - 1)\log_2 n$. However, we may also consider this family to be a family of t to 1 strongly universal hash function as proposed in [22].

6.4 Compressing Arbitrary Length Messages

One of the main properties of hash functions is that they should be applied to any argument of any size. Damgard suggested a design principle in [8] based on fixed size collision free hash functions. Another method has been appeared in [31] and is the dual of Blum-Micali pseudorandom bit generator (let us call it ZMI method). We show that using the perfect one-way permutation proposed in Theorem 4, these two methods actually yield one scheme for hashing long messages.

Damgard's design principle: Let $l(n)$ be a polynomial with $l(n) > n$, let f be a collision free one-way hash function $f : \Sigma^{n+t} \to \Sigma^n$, $\alpha \in_r \Sigma^n$, split $l(n)$-bit message x into t bit blocks, let the blocks be denoted by $x_1, x_2, \ldots, x_{\frac{l(n)}{t}}$. Let

$$\begin{aligned} y_0 &= \alpha \\ &\vdots \\ y_{i+1} &= f(y_i \parallel x_{i+1}) \end{aligned}$$

then $h(x) = y_{\frac{l(n)}{t}}$ would be the hash value of the long message x.

ZMI method: Let f be a one-way permutation $f : \Sigma^{n+t} \to \Sigma^{n+t}$, let $I(n) = (i_1, i_2, \ldots, i_t)$ denote the known simultaneously hard bits of f, let $x = x_t \ldots x_2 x_1 \in \Sigma^t$, $b \in \Sigma^n$, define:

$$\text{ins}_{I(n)}(b, x) = \ldots b_i x_1 b_{i-1} \ldots b_2 b_1$$

Let $z \in \Sigma^{n+t}$, denote by $\text{drop}_{I(n)}(z)$ a function dropping the i_1-th, ..., i_t-th bits of z. Let l be a polynomial with $l(n) > n$, $\alpha \in \Sigma^n$, split $l(n)$-bit message x into t bit blocks be denoted by

$x_1, x_2, \ldots, x_{\frac{l(n)}{t}}$, where $x_i \in \Sigma^t$ for each $1 \leq i \leq \frac{l(n)}{t}$. Let h be the function from $\Sigma^{l(n)}$ to Σ^n defined by:

$$\begin{aligned} y_0 &= \alpha \\ y_1 &= \mathrm{drop}_{I(n)}(f(\mathrm{ins}_{I(n)}(y_0, x_{\frac{l(n)}{t}}))) \\ &\vdots \\ y_i &= \mathrm{drop}_{I(n)}(f(\mathrm{ins}_{I(n)}(y_{i-1}, x_{\frac{l(n)}{t}-i+1}))) \end{aligned}$$

then $h(x) = y_{\frac{l(n)}{t}} = \mathrm{drop}_{I(n)}(f(\mathrm{ins}_{I(n)}(y_{\frac{l(n)}{t}-1}, x_1)))$. (In the original ZMI scheme $h(x) = f(\mathrm{ins}_{I(n)}(y_{\frac{l(n)}{t}-1}, x_1)))$. If we use the strong one-way permutation m in ZMI scheme for f, since the t least significant bits are simultaneously hard bits, then $\mathrm{drop}_{I(n)}$ function performs identically to the chop_t function defined in Corollary 5. So, $\mathrm{drop}_{I(n)}(f(x))$ in ZMI method would be identical to $\mathrm{chop}_t(m(x))$ of Corollary 5, which is a universal one-way hash function from Σ^{n+t} to Σ^n. On the other hand, when t last bits of a function are simultaneously hard bits, then $\mathrm{ins}_{I(n)}(y_0, x_{\frac{l(n)}{t}})$ would yield the same result as $(y_0 \parallel x_{\frac{l(n)}{t}})$. So, practicing the strong one-way permutation with ZMI scheme, would yield the same result as either practicing the one-way hash function proposed in Corollary 5 with Damgard's design principle, when the message blocks are fed in a similar order.

7 A Single construction for PBG and UOWHF

Each iteration of the pseudorandom bit generator presented in Theorem 5 is identical to the hash function presented in Corollary 5. Assume that we have a computational resource for at most k=63 bits. For the construction of the PBG of Theorem 5, an algorithm should extract at most $n-k-1$ bits, and throw away at least $k+1$ bits on each iteration. On the other hand, for the construction of the one-way hash function according to Corollary 5, we may chop at most $n-k-1$ bits, and leave $k+1$ bits as the hash value. If we choose $k < t < n-k$, for example for $n = 512$ we choose $64 \leq t \leq 448$, then the algorithm can be used both for pseudorandom bit generation and universal one-way hashing.

8 Conclusions and Extensions

1. We constructed a strong permutation with a $k+1$-bit perfect permutation, namely a complete permutation whose $k+1$ output bits are independent. A $k+1$-bit perfect permutation can be constructed easily as follows:

   $$v(x) = c(x) \oplus PBG_{k+1}(x_l, \ldots, x_1)$$

   where $x \in \Sigma^n$, and $c(x)$ is a complete permutation, and $PBG_{k+1}(x_l, \ldots, x_1)$ denotes $k+1$ output bits of a pseudorandom bit generator where the seed is $l > k$ bits of the x. Then, we constructed a UOWHF and also an efficient pseudorandom bit generator with the strong permutation. This confirms Naor and Yung's conjecture [22] that if pseudorandom bit generators exist then UOWHF exist.

2. For the construction of the strong permutation we assumed that the position of $k+1$ hard bits of the one-way function f complies with $k+1$ independent bits of v. The following generalisation can easily be shown to be true.

If v is a perfect permutation then $m = f \circ v \circ f$ is a strong one-way permutation, where f is a one way permutation.

In other words, there is no need to know the exact positions of hard bits of f. As we mentioned earlier, the running time of a perfect permutation based on DES structure for large enough n, e.g., n=512, is rather big.

A reasonable question is whether we can apply some simpler mathematical functions or a compositions of such functions for v, for example $y = (a_i x)^3 \bmod m$, and/or $y = (ax + b) \bmod m$.

ACKNOWLEDGMENT

We would like to thank the members of the CCSR for their support and assistance during the preparation of this work.

References

[1] Selim G. Akl. On the security of compressed encoding. In *Advances in Cryptology - CRYPTO '83*, pages 209–230. Plenum Publishing Corporation, 1983.

[2] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, 1988.

[3] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[4] L. Brown. A proposed design for an extended DES. In *Computer Security in the Age of Information*. North-Holland, 1989. Proceedings of the Fifth IFIP International Conference on computer Security, IFIP/Sec '88.

[5] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[6] D. Coppersmith. Analysis of ISO/CCITT Document X.509 Annex D, 1989.

[7] I. B. Damgard. Collision free hash functions and public key signature schemes. In *Advances in Cryptology - EUROCRYPT '87*, volume 304 of *Lecture Notes in Computer Science*, pages 203–216. Springer-Verlag, 1987.

[8] I. B. Damgard. A design principle for hash functions. In *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1989.

[9] D. E. Denning. Digital signatures with RSA and other public-key cryptosystems. *Communications of the ACM*, 27(4):388–392, 1984.

[10] M. Girault. Hash-functions using modulo-n operations. In *Advances in Cryptology - EUROCRYPT '87*, volume 304 of *Lecture Notes in Computer Science*, pages 218–226. Springer-Verlag, 1987.

[11] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[12] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *the 21st ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[13] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *the 21st ACM Symposium on Theory of Computing*, pages 12–24, 1989.

[14] W. De Jonge and D. Chaum. Attacks on some RSA signatures. In *Advances in Cryptology - CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 18–27. Springer-Verlag, 1985.

[15] W. De Jonge and D. Chaum. Some variations on RSA signatures and their security. In *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 49–59. Springer-Verlag, 1986.

[16] R. R. Jueneman. Electronic document authentication. *IEEE Network Magazine*, 1(2):17–23, 1987.

[17] J. B. Kam and G. I. Davida. Structured design of substitution-permutation encryption networks. *IEEE Transactions on Computers*, 28(10):747–753, 1979.

[18] S. M. Matyas, C. H. Meyer, and J. Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27(10A):5658–5659, 1985.

[19] R. C. Merkle. One way hash functions and DES. In *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, 1989.

[20] S. Miyaguchi, K. Ohta, and M. Iwata. Confirmation that some hash functions are not collision free. In *Abstracts of EUROCRYPT '90*, pages 293–308, 1990.

[21] J. H. Moore. Protocol failures in cryptosystems. *Proceedings of the IEEE*, 76(5):594–601, 1988.

[22] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *the 21st ACM Symposium on Theory of Computing*, pages 33–43, 1989.

[23] J. Quisquater and M. Girault. 2n-bit hash functions using n-bit symmetric block cipher algorithms. In *Abstracts of EUROCRYPT '89*, 1989.

[24] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *the 22nd ACM Symposium on Theory of Computing*, pages 387–394, 1990.

[25] A. De Santis and M. Yung. On the design of provably-secure cryptographic hash functions. In *Abstracts of EUROCRYPT '90*, pages 377–397, 1990.

[26] A. F. Webster and S. E. Tavares. On the design of S-boxes. In *Advances in Cryptology - CRYPTO '85*, Lecture Notes in Computer Science, pages 523–534. Springer-Verlag, 1985.

[27] M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

[28] R. S. Winternitz. Producing a one-way hash function from DES. In *Advances in Cryptology - CRYPTO '83*, pages 203–207. Plenum Publishing Corporation, 1983.

[29] R. S. Winternitz. A secure one-way hash function built from DES. In *the 1984 IEEE Symposium on Security and Privacy*, 1984.

[30] A. C. Yao. Theory and applications of trapdoor functions. In *the 23rd IEEE Symposium on the Foundations of Computer Science*, pages 80–91, 1982.

[31] Y. Zheng, T. Matsumoto, and H. Imai. Duality between Two Cryptographic Primitives. In *the 8-th International Conference on Applied Algebra, Algebraic Algorithms and Error Correcting Codes*, page 15, 1990.

[32] Y. Zheng, T. Matsumoto, and H. Imai. Structural properties of one-way hash-functions. In *CRYPTO '90*, pages 263–280, 1990.