Duality between Two Cryptographic Primitives

Yuliang Zheng
Tsutomu Matsumoto
Hideki Imai

Division of Electrical and Computer Engineering
Yokohama National University
156 Tokiwadai, Hodogaya, Yokohama, 240 JAPAN

Abstract

This paper reveals a duality between constructions of two basic cryptographic primitives, *pseudo-random string generators* and *one-way hash functions*. Applying the duality, we present a construction for *universal one-way hash functions* assuming the existence of one-way permutations. Under a stronger assumption, the existence of distinction-intractable permutations, we prove that the construction constitutes a *collision-intractable hash function*. Using ideas behind the construction, we propose *practical* one-way hash functions, the fastest of which compress nearly $2n$-bit long input into n-bit long output strings by applying only *twice* a one-way function.

1 Introduction

Pseudo-random string generators and one-way hash functions are two basic cryptographic primitives. Informally, a *pseudo-random string generator* is a function that on input a random string called a *seed* outputs a longer string which can not be efficiently distinguished from a truly random one. In contrast, a *one-way hash function* outputs a short string on input a long one, with the property that it is computationally difficult to find a pair of strings that are compressed into the same one. In a sense, pseudo-random string generators and one-way hash functions behave in a *dual* fashion.

It has been proved that pseudo-random string generators can be constructed under the assumption of the existence of one-way functions [ILL89]. However, at the time when this paper was written, the best known result on the construction of (universal) one-way hash functions was based on the assumption of the existence of one-way quasi-injections [ZMI90a]. (See also [ZMI90b].) The aim of this research is to explore the intuition that there is a duality between pseudo-random string generators and one-way hash functions, and to apply techniques developed for the former to the construction of the latter under weaker assumptions. Though our aim has not been achieved yet, we obtain a new construction for (universal) one-way hash functions assuming the existence of one-way permutations. Using the construction, we design hash functions that are very efficient and seem to be also one-way.

The paper is organized as follows. In Section 2, we introduce notions and notation. In Section 3, we discuss a duality between the construction of pseudo-random string generators and that of one-way hash functions. Applying the duality, we present in Section 4 a construction for *universal one-way hash functions* assuming the existence of one-way permutations. Under a stronger assumption, the existence of distinction-intractable permutations, we prove in section 5 that the construction constitutes a *collision-intractable hash function*. In Section 6, we use ideas behind the construction to design *practical* (supposed) one-way hash functions. The fastest of these functions compress nearly $2n$-bit long input into n-bit long output strings by applying only *twice* a one-way function.

2 Terminology and Preliminaries

The set of all positive integers is denoted by $\mathbf{N}$. Let $\Sigma = \{0,1\}$ be the alphabet we consider. For $n \in \mathbf{N}$, denote by Σ^n the set of all strings over Σ with length n, by Σ^* that of all finite length strings including the empty string, denoted by λ, over Σ, and by Σ^+ the set $\Sigma^* - \{\lambda\}$. The concatenation of two strings x, y is denoted by $x \diamond y$, or simply by xy if no confusion arises. When $x, y \in \Sigma^n$, the bit-wise $\bmod 2$ addition, also called the exclusive-or (XOR), of x and y is denoted by $x \oplus y$. The length of a string x is denoted by $|x|$, and the number of elements in a set S is denoted by $\sharp S$.

Let ℓ be a monotone increasing function from $\mathbf{N}$ to $\mathbf{N}$, and f a (total) function from D to R, where $D = \bigcup_n D_n, D_n \subseteq \Sigma^n$, and $R = \bigcup_n R_n, R_n \subseteq \Sigma^{\ell(n)}$. D is called the *domain*, and R the *range* of f. In this paper it is assumed, unless otherwise specified, that $D_n = \Sigma^n$ and $R_n = \Sigma^{\ell(n)}$. Denote by f_n the restriction of f on Σ^n. We are concerned only with the case when the range of f_n is $\Sigma^{\ell(n)}$, i.e., f_n is a function from Σ^n to $\Sigma^{\ell(n)}$. f is an *injection* if each f_n is a one-to-one function, and is a *permutation* if each f_n is a one-to-one and onto function. f is (deterministic/probabilistic) *polynomial time computable* if there is a (deterministic/probabilistic) polynomial time algorithm (Turing machine) computing $f(x)$ for all $x \in D$. The composition of two functions f and g is defined as $f \circ g(x) = f(g(x))$. In particular, the i-fold composition of f is denoted by $f^{(i)}$.

A (probability) *ensemble* E with length $\ell(n)$ is a family of *probability distributions* $\{E_n | E_n : \Sigma^{\ell(n)} \to [0,1], n \in \mathbf{N}\}$. The *uniform ensemble* U with length $\ell(n)$ is the family of *uniform probability distributions* U_n, where each U_n is defined as $U_n(x) = 1/2^{\ell(n)}$ for all $x \in \Sigma^{\ell(n)}$. By $x \in_E \Sigma^{\ell(n)}$ we mean that x is randomly chosen from $\Sigma^{\ell(n)}$ 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 a (probabilistic) algorithm M that on input n outputs an $x \in_E \Sigma^{\ell(n)}$, and *polynomially samplable* if furthermore, the running time of M is polynomially bounded.

2.1 Pseudo-random String Generators and One-Way Functions

Let ℓ be a polynomial. A *statistical test* is a probabilistic polynomial time algorithm T that, on input a string x, outputs a bit 0/1. Let E^1 and E^2 be ensembles both with length $\ell(n)$. 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, $|\Pr\{T(x_1) = 1\} - \Pr\{T(x_2) = 1\}| < 1/Q(n)$, where $x_1 \in_{E^1} \Sigma^{\ell(n)}, x_2 \in_{E^2} \Sigma^{\ell(n)}$. A polynomially samplable ensemble E is *pseudo-random* if it is indistinguishable from the uniform ensemble U with the same length.

Now we further assume that ℓ is a polynomial with $\ell(n) > n$. A *string generator* extending n-bit input into $\ell(n)$-bit output strings is a deterministic polynomial time computable function $g : D \to R$ where $D = \bigcup_n \Sigma^n$ and $R = \bigcup_n \Sigma^{\ell(n)}$. g will be denoted also by $g = \{g_n \mid n \in \mathbf{N}\}$. Let $g_n(U)$ be the probability distribution defined by the random variable $g_n(x)$ where $x \in_R \Sigma^n$, and let $g(U) = \{g_n(U) \mid n \in \mathbf{N}\}$. Clearly, $g(U)$ is polynomially samplable. The following definition can be found in [Yao82] (see also [BM84], [GGM86] and [ILL89]).

Definition 1 *$g = \{g_n \mid n \in \mathbf{N}\}$ is a* (cryptographically secure) pseudo-random string generator (PSG) *if $g(U)$ is pseudo-random.*

One-way function is the basis of most of modern cryptographic functions and protocols [IL89]. The following definition is from [ILL89].

Definition 2 *Let $f : D \to R$, where $D = \bigcup_n \Sigma^n$ and $R = \bigcup_n \Sigma^{\ell(n)}$, be a polynomial time computable function, and let E be an ensemble with length n. We say that (1) f is* one-way with respect to E *if for each probabilistic polynomial time algorithm M, for each polynomial Q and for all sufficiently large n, $\Pr\{f_n(x) = f_n(M(f_n(x)))\} < 1/Q(n)$, when $x \in_E D_n$. (2) f is* one-way *if it is one-way with respect to the uniform ensemble U with length n.*

We note that a function f is one-way (with respect to the uniform ensemble U with length n) iff f is one-way with respect to *all* pseudo-random ensembles with the same length. This fact will be used in the proof for Theorem 2. Next we introduce the concept of (simultaneously) hard bits.

Definition 3 *Assume that $f : D \to R$ is a one-way function, where $D = \bigcup_n \Sigma^n$ and $R = \bigcup_n \Sigma^{\ell(n)}$. Also assume that $i_1, i_2, \ldots, i_t$ are functions from $\mathbf{N}$ to $\mathbf{N}$, with $1 \le i_j(n) \le n$ for each $1 \le j \le t$. Denote by E^1_n and E^2_n the probability distributions defined by the random variables $x_{i_t(n)} \cdots x_{i_2(n)} x_{i_1(n)} \diamond f(x)$ and $r_t \cdots r_2 r_1 \diamond f(x)$ 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 $E^1 = \{E^1_n \mid n \in \mathbf{N}\}$ and $E^2 = \{E^2_n \mid n \in \mathbf{N}\}$. We say that (1) $i_1(n)$ is a* hard bit *of f if for each probabilistic polynomial time algorithm M, for each polynomial Q and for all sufficiently large n, $\Pr\{M(f_n(x)) = x'_{i_1(n)}\} < 1/2 + 1/Q(n)$, where $x \in_R \Sigma^n$ and $x'_{i_1(n)}$ is the $i_1(n)$-th bit of an $x' \in \Sigma^n$ satisfying $f(x) = f(x')$. (2) $i_1(n), i_2(n), \ldots, i_t(n)$ are* simultaneously *hard bits of f if E^1 and E^2 are indistinguishable from each other.*

2.2 One-Way Hash Functions

There are basically two kinds of one-way hash functions: *universal one-way hash functions* and *collision-intractable hash functions* (or shortly UOHs and CIHs, respectively). In [Mer89] the former is called *weakly* and the latter *strongly*, one-way hash functions respectively. Naor and Yung gave a formal definition for UOH [NY89], and Damgård gave for CIH [Dam89]. The definition for UOH to be given below is from [ZMI90a] [ZMI90b] in which many other results, such as a construction for UOHs assuming the existence of one-way quasi-injections, are presented.

Let ℓ and m be polynomials with $\ell(n) > m(n)$, H be a family of functions defined by $H = \bigcup_n H_n$ where H_n is a (possibly multi-)set of functions from $\Sigma^{\ell(n)}$ to $\Sigma^{m(n)}$. Call H a *hash function* compressing $\ell(n)$-bit input into $m(n)$-bit output strings. For two strings $x, y \in \Sigma^{\ell(n)}$ 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 (in n) time algorithm computing all $h \in H$, and *accessible* if there is a probabilistic polynomial time algorithm that on input $n \in \mathbf{N}$ outputs uniformly at random a description of $h \in H_n$. All hash functions considered in this paper are both polynomial time computable and accessible.

Let H be a hash function compressing $\ell(n)$-bit input into n-bit output strings, and E an ensemble with length $\ell(n)$. The definition for UOH is best described as a three-party game. The three parties are S (an *initial-string supplier*), G (a *hash function instance generator*) and F (a *collision-string finder*). S is an oracle whose power is un-limited, and both G and F are probabilistic polynomial time algorithms. The first move is taken by S, who outputs an *initial-string* $x \in_E \Sigma^{\ell(n)}$ and sends it to both G and F. The second move is taken by G, who chooses, independently of x, an $h \in_R H_n$ and sends it to F. The third and also final (null) move is taken by F, who on input $x \in \Sigma^{\ell(n)}$ and $h \in H_n$ outputs either "?" (I don't know) or a string $y \in \Sigma^{\ell(n)}$ such that $x \neq y$ and $h(x) = h(y)$. F wins a game iff his/her output is *not* equal to "?". Informally, H is a universal one-way hash function with respect to E if for any collision-string finder F, the probability that F wins a game is negligible. More precisely:

Definition 4 *Let H be a hash function compressing $\ell(n)$-bit input into n-bit output strings, P a collection of ensembles with length $\ell(n)$, and F a collision-string finder. H is a* universal one-way hash function with respect to P*, denoted by UOH/P, if for each $E \in P$, for each F, for each polynomial Q, and for all sufficiently large n,* $\Pr\{F(x, h) \neq ?\} < 1/Q(n)$*, where x and h are independently chosen from $\Sigma^{\ell(n)}$ and H_n according to E_n and to the uniform distribution over H_n respectively, and the probability* $\Pr\{F(x, h) \neq ?\}$ *is computed over $\Sigma^{\ell(n)}$, H_n and the sample space of all finite strings of coin flips that F could have tossed.*

If E is an ensemble with length $\ell(n)$, UOH/E is synonymous with UOH/$\{E\}$. In this paper we only consider one version of UOH that is denoted by UOH/$EN[\ell]$, where $EN[\ell]$ is the collection of all ensembles with length $\ell(n)$. Other versions such as UOH/$PSE[\ell]$ and UOH/U are also of interest, where $PSE[\ell]$ is the collection of all polynomially samplable ensembles and U is the uniform ensemble, all with length $\ell(n)$. Relationships among various versions of one-way hash functions including UOH/$EN[\ell]$, UOH/$PSE[\ell]$, UOH/U

and CIH are discussed in [ZMI90a] [ZMI90b]. Of the results obtained in the two papers the most important is that *one-way hash functions in the sense of UOH/EN*$[\ell]$ *exist iff those in the sense of UOH/U exist.*

We end this section with a definition for CIH that corresponds to *collision free function family* given in [Dam89]. Let A, a *collision-pair finder*, be a probabilistic polynomial time algorithm that on input $h \in H_n$ outputs either "?" or a pair of strings $x, y \in \Sigma^{\ell(n)}$ with $x \neq y$ and $h(x) = h(y)$.

Definition 5 *H is called a* collision-intractable hash function (CIH) *if for each A, for each polynomial Q, and for all sufficiently large n,* $\Pr\{A(h) \neq ?\} < 1/Q(n)$, *where $h \in_R H_n$, and the probability* $\Pr\{A(h) \neq ?\}$ *is computed over H_n and the sample space of all finite strings of coin flips that A could have tossed.*

3 Extending and Compressing Methods

In this section we discuss a duality between the construction of pseudo-random string generators and that of one-way hash functions. Throughout this section, t and s are integers, $\ell_0, \ell_1, \ldots, \ell_s$ are polynomials in n with $\ell_0(n) = n$ and $\ell_i(n) < \ell_{i+1}(n)$, and k is a polynomial in n such that $t \cdot k(n) > n$. Denote by ℓ the polynomial $t \cdot k$.

3.1 Serial Versions

Two extending and two compressing methods which are serial in nature are introduced below. Lemma 1 (serial-extending 1) is the dual of Lemma 3 (serial-compressing 1), and Lemma 2 (serial-extending 2) is that of Lemma 4 (serial-compressing 2).

3.1.1 Serial Extending Lemmas

For each $0 \leq i \leq s-1$, let $g^i = \{g_n^i \mid n \in \mathbf{N}\}$ be a PSG extending $\ell_i(n)$-bit input into $\ell_{i+1}(n)$-bit output strings. The following lemma is a direct consequence of the definition for PSG.

Lemma 1 (serial-extending 1) *Let $g = \{g_n = g_n^{s-1} \circ g_n^{s-2} \circ \cdots \circ g_n^0 \mid n \in \mathbf{N}\}$. Then g is a PSG extending n-bit input into $\ell_s(n)$-bit output strings.*

A PSG extending n-bit input into $(n+t)$-bit output strings is called a t-extender. Let y be a finite length string. Denote by $\mathrm{head}_i(y)$ the first i bits and by $\mathrm{tail}_i(y)$ the last i bits of y. The following lemma is due to Boppana and Hirschfeld [BH89].

Lemma 2 (serial-extending 2) *Let $e = \{e_n \mid n \in \mathbf{N}\}$ be a t-extender. For an n-bit string x, let $b_i(x) = \mathrm{head}_t \circ e_n \circ (\mathrm{tail}_n \circ e_n)^{(i-1)}(x)$, where $1 \leq i \leq k(n)$. Let g_n be the function defined by $g_n(x) = b_{k(n)}(x) \diamond \cdots \diamond b_2(x) \diamond b_1(x)$. Then $g = \{g_n \mid n \in \mathbf{N}\}$ is a PSG extending n-bit input into $\ell(n)$-bit output strings.*

3.1.2 Serial Compressing Lemmas

For each $1 \leq i \leq s$, let $H^i = \bigcup_n H_n^i$ be a UOH/$EN[\ell_i]$ (or CIH) compressing $\ell_i(n)$-bit input into $\ell_{i-1}(n)$-bit output strings. Naor and Yung proved the following serial compressing lemma [NY89].

Lemma 3 (serial-compressing 1) *Let $H = \bigcup_n H_n$, where $H_n = \{h = h_1 \circ h_2 \circ \cdots \circ h_s \mid h_i \in H_n^i, 1 \leq i \leq s\}$. Then H is a UOH/$EN[\ell_s]$ (or CIH) compressing $\ell_s(n)$-bit input into n-bit output strings.*

Let $\ell'(n) = n + t$, and let $C = \bigcup_n C_n$ be a UOH/$EN[\ell']$ (or CIH) compressing $\ell'(n)$-bit input into n-bit output strings. Such a UOH/$EN[\ell']$ (or CIH) is called a t-compressor. Then we have the following lemma that is a restricted version of Theorem 3.1 of [Dam89]. The main idea behind the lemma also appeared in [Mer89], where it was called the "meta-method".

Lemma 4 (serial-compressing 2) *Let $C = \bigcup_n C_n$ be a t-compressor. For each $c \in C_n$ and each $\alpha \in \Sigma^n$, let $h_{c,\alpha}$ be the function defined by $h_{c,\alpha}(x) = c(\cdots(c(\alpha \diamond b_{k(n)}(x)) \diamond b_{k(n)-1}(x))\cdots \diamond b_1(x))$, where $x = x_{\ell(n)} \cdots x_2 x_1$ is an $\ell(n)$-bit string and $b_i(x) = x_{t(i-1)+t} \cdots x_{t(i-1)+2} x_{t(i-1)+1}$. Let $H_n = \{h_{c,\alpha} | c \in C_n, \alpha \in \Sigma^n\}$, and $H = \bigcup_n H_n$. Then H is a UOH/$EN[\ell]$ (or CIH) compressing $\ell(n)$-bit input into n-bit output strings.*

3.2 Parallel Versions

In their nice paper [GGM86], Goldreich et al. presented a method for constructing *pseudo-random functions* from pseudo-random string generators. Their construction provides us a configuration for generating pseudo-random strings in parallel, when given polynomially many processors. This observation is the very basis of Micali and Schnorr's parallel PSGs [MSc88]. On the other hand, a parallel hashing method was considered in several papers such as [WC81], [NY89] and [Dam89]. Duality between these two methods is clear. Details are omitted here.

4 PSGs and UOHs from One-Way Permutations

Throughout this section, it is assumed that f is a one-way permutation on $D = \bigcup_n \Sigma^n$, and that $i(n)$ has been proved to be a hard bit of f.

4.1 PSGs from One-Way Permutations

Denote by $\text{extract}_i(x)$ a function extracting the i-th bit of $x \in \Sigma^n$. The following theorem is due to Blum and Micali [BM84].

Theorem 1 *Let ℓ be a polynomial with $\ell(n) > n$, and let g_n be the function defined by $g_n(x) = b_{\ell(n)}(x) \cdots b_2(x) b_1(x)$ where $x \in \Sigma^n$ and $b_j(x) = \text{extract}_{i(n)}(f_n^{(j)}(x))$. Then under the assumption that f is a one-way permutation, $g = \{g_n | n \in \mathbf{N}\}$ is a PSG extending n-bit input into $\ell(n)$-bit output strings.*

The efficiency of g can be improved by changing $\mathrm{extract}_{i(n)}()$ to a function that extracts all known simultaneously hard bits of f.

4.2 UOHs from One-Way Permutations

For $b \in \Sigma$, $x \in \Sigma^{n-1}$ and $y \in \Sigma^n$, define $\mathrm{ins}_i(x, b) = x_{n-1}x_{i-2}\cdots x_i b x_{i-1}\cdots x_2 x_1$, and denote by $\mathrm{drop}_i(y)$ a function dropping the i-th bit of y. As the dual of Theorem 1, we have the following result.

Theorem 2 *Let ℓ be a polynomial with $\ell(n) > n$, $\alpha \in \Sigma^{n-1}$ and $x = x_{\ell(n)}\cdots x_2 x_1$ where $x_i \in \Sigma$ for each $1 \le i \le \ell(n)$. Let h_α be the function from $\Sigma^{\ell(n)}$ to Σ^n defined by: $y_0 = \alpha, y_1 = \mathrm{drop}_{i(n)}(f_n(\mathrm{ins}_{i(n)}(y_0, x_{\ell(n)}))), \cdots, y_j = \mathrm{drop}_{i(n)}(f_n(\mathrm{ins}_{i(n)}(y_{j-1}, x_{\ell(n)-j+1}))), \cdots,$ $h_\alpha(x) = f_n(\mathrm{ins}_{i(n)}(y_{\ell(n)-1}, x_1))$. Let $H_n = \{h_\alpha \mid \alpha \in \Sigma^{n-1}\}$ and $H = \bigcup_n H_n$. Then under the assumption that f is a one-way permutation, H is a UOH/EN$[\ell]$ compressing $\ell(n)$-bit input into n-bit output strings.*

Proof : Assume that E is an initial-string ensemble and F a collision-string finder. Let $x \in_E \Sigma^{\ell(n)}$. We show that if F finds with probability $p(n)$ a string x' such that $h(x) = h(x')$ where $h \in_R H_n$, then there is a probabilistic polynomial time algorithm M that finds, with probability greater than $p(n)/(\ell(n)-1)$, the inverse $f_n^{-1}(w)$ of an n-bit string $w \in_T \Sigma^n$, where T is a pseudo-random ensemble.

The proving procedure consists of three parts. In the first part, we show that every execution of f_n in h defines two pseudo-random ensembles S^j and R^j, if $\alpha \in_R \Sigma^{n-1}$. From each S^j we construct another pseudo-random ensemble T^j. In the second part, we construct a probabilistic polynomial time algorithm M. M first obtains $w \in_T \Sigma^n$, where T is an ensemble chosen uniformly at random from all ensembles T^j. Then it computes the inverse $f_n^{-1}(w)$ by the use of the collision-string finder F. In the third part we estimate the probability that M outputs $f_n^{-1}(w)$ correctly.

Let S_n^1 be the probability distribution defined by $f_n(\mathrm{ins}_{i(n)}(\alpha, x_{\ell(n)}))$, where $\alpha \in_R \Sigma^{n-1}$ and $x_{\ell(n)}$ is the last bit of $x \in_E \Sigma^{\ell(n)}$. Let $S^1 = \{S_n^1 \mid n \in \mathbf{N}\}$. Clearly S^1 is polynomially samplable (when $x \in_E \Sigma^{\ell(n)}$ is given). Now we show that S^1 is indistinguishable from the uniform ensemble.

Let A be a probabilistic polynomial time algorithm. For $v \in \Sigma$, denote by $\Pr^v$ the probability that A outputs 1 on input $f_n(\mathrm{ins}_{i(n)}(\alpha, v))$. Since $\alpha \in_R \Sigma^{n-1}$ and $i(n)$ is a hard bit of f (by assumption), we can think of $z = f_n(\mathrm{ins}_{i(n)}(\alpha, x_{\ell(n)}))$ as a probabilistic encryption of $x_{\ell(n)}$ [GM84]. Thus for any probabilistic polynomial time algorithm A, for any polynomial Q, for all sufficiently large n, we have $|\Pr^0 - \Pr^1| < 1/Q(n)$. Now let $v \in_R \Sigma$. Then $\Pr^v = \Pr^0 \cdot \Pr\{v = 0\} + \Pr^1 \cdot \Pr\{v = 1\} = (\Pr^0 + \Pr^1)/2$, and hence $|\Pr^v - \Pr^0| = |\Pr^v - \Pr^1| = |\Pr^0 - \Pr^1|/2 < 1/2Q(n)$. This implies that $|\Pr^v - \Pr^{x_{\ell(n)}}| < 1/2Q(n)$, no matter how $x_{\ell(n)}$ is chosen from Σ. Note that when $\alpha \in_R \Sigma^{n-1}$ and $v \in_R \Sigma$, we have $f_n(\mathrm{ins}_{i(n)}(\alpha, v)) \in_R \Sigma^n$, since f is a permutation. From these discussions, we see that S^1 is indeed pseudo-random.

Let $R^1 = \{R_n^1 \mid n \in \mathbf{N}\}$, where R_n^1 is defined by $\mathrm{drop}_{i(n)}(f_n(\mathrm{ins}_{i(n)}(\alpha, x_{\ell(n)})))$. Then R^1 is also pseudo-random.

Let $S^2 = \{S_n^2 \mid n \in \mathbf{N}\}$, where S_n^2 is defined by $f_n(\mathrm{ins}_{i(n)}(\beta, x_{\ell(n)-1}))$, $\beta \in_{R^1} \Sigma^{n-1}$ and $x_{\ell(n)-1}$ is the second last bit of $x \in_E \Sigma^{\ell(n)}$. Then S^2 is also pseudo-random, for otherwise there would be a statistical test distinguishing R^1 from the uniform ensemble with length $n-1$. Similarly, for each $2 < j \leq \ell(n)$, we can define S^j, R^j and prove that they are both pseudo-random.

For each S^j, $1 \leq j \leq \ell(n) - 1$, define T^j as follows: $w \in_{T^j} \Sigma^n$ is produced by first generating a string $w' \in_{S^j} \Sigma^n$, then reversing the $i(n)$-th bit of w', i.e., $w = w' \oplus 0^{n-i(n)}10^{i(n)-1} = f_n(\mathrm{ins}_{i(n)}(y_{j-1}, x_{\ell(n)-j+1})) \oplus 0^{n-i(n)}10^{i(n)-1}$, where 0^i denotes the all-0 string of length i and $\oplus$ the bit-wise exclusive-or operation on two strings. Obviously, T^j is also pseudo-random.

We are ready to describe the algorithm M. Let O be an oracle that on input $n \in \mathbf{N}$ outputs a string $x \in_E \Sigma^{\ell(n)}$.

Algorithm M:

1. Choose $\alpha \in_R \Sigma^{n-1}$.
2. Query the oracle O with n. Let the answer by O be x. Note that $x \in_E \Sigma^{\ell(n)}$.
3. Choose at random a $1 \leq k \leq \ell(n) - 1$, and let $T = T^k$.
4. Let $w = f_n(\mathrm{ins}_{i(n)}(y_{k-1}, x_{\ell(n)-k+1})) \oplus 0^{n-i(n)}10^{i(n)-1}$, i.e., choose a $w \in_T \Sigma^n$.
5. Query the collision-string finder F with n, h and x. If F finds a string x' such that $h(x) = h(x')$, then output $\mathrm{ins}_{i(n)}(y'_{k-1}, x'_{\ell(n)-k+1})$, where y'_{k-1} and $x'_{\ell(n)-k+1}$ are defined similarly to y_{k-1} and $x_{\ell(n)-k+1}$ respectively. Otherwise, output a $z \in_R \Sigma^n$.

The running time of M is clearly bounded by a polynomial in n. Now we estimate the probability that the algorithm M outputs $f_n^{-1}(w)$ correctly. Note that when F finds an x' such that $h(x) = h(x')$, then there is an $1 \leq m \leq \ell(n) - 1$ such that $\mathrm{ins}_{i(n)}(y_{m-1}, x_{\ell(n)-m+1}) \neq \mathrm{ins}_{i(n)}(y'_{m-1}, x'_{\ell(n)-m+1})$ and $\mathrm{ins}_{i(n)}(y_{j-1}, x_{\ell(n)-j+1}) = \mathrm{ins}_{i(n)}(y'_{j-1}, x'_{\ell(n)-j+1})$ for each $m < j \leq \ell(n)$. Since k is chosen independently of F, the probability that $k = m$ is $1/(\ell(n) - 1)$. So the probability that M outputs $f_n^{-1}(w)$ correctly is $\Pr\{M \text{ outputs } f_n^{-1}(w)\} > \Pr\{M \text{ outputs } f_n^{-1}(w) \mid F \text{ finds } x'\} \cdot \Pr\{F \text{ finds } x'\} = p(n)/(\ell(n) - 1)$.

When $p(n) \geq 1/Q(n)$ for some polynomial Q, i.e., H is not a one-way hash function in the sense of UOH/$EN[\ell]$, we have $\Pr\{M \text{ outputs } f_n^{-1}(w)\} > 1/Q'$ where Q' is the polynomial defined by $Q'(n) = Q(n)(\ell(n) - 1)$. In other words, f is not one-way with respect to the pseudo-random ensemble T, hence not one-way with respect to the uniform ensemble U (see the note following Definition 2). This is a contradiction and the proof is completed. □

Now let $I(n) = \{i_1, i_2, \ldots, i_{t(n)}\}$ be known simultaneously hard bits of f with $t(n) = O(\log n)$. Let $b = b_{t(n)} \cdots b_2 b_1 \in \Sigma^{t(n)}$, $x \in \Sigma^{n-t(n)}$ and $y \in \Sigma^n$. Define $\mathrm{ins}_{I(n)}(x, b) = x_{n-t(n)} \cdots x_{i_{t(n)}} b_{t(n)} x_{i_{t(n)}-1} \cdots x_{i_1} b_1 x_{i_1-1} \cdots x_2 x_1$, and denote by $\mathrm{drop}_{I(n)}(y)$ a function dropping the i_1-th, i_2-th, …, $i_{t(n)}$-th bits of y. Then by changing $\mathrm{drop}_{i(n)}()$ to $\mathrm{drop}_{I(n)}()$, and x_i to $x_{t(n)(i-1)+t(n)} \cdots x_{t(n)(i-1)+2} x_{t(n)(i-1)+1}$, the above constructed H is improved to a hash function that compresses $t(n)\ell(n)$-bit input into n-bit output strings.

5 CIHs from Distinction-Intractable Permutations

We were not able to prove that the hash function H constructed in Section 4.2 is also a CIH. Under a stronger assumption to be stated below, H can be proved to be indeed a CIH.

Assume that $f : D \to R$ is a polynomial time computable function. f is *distinction-intractable* at the $i(n)$-th bit if it is computationally difficult to find a pair of strings $x, y \in D_n$ such that $f_n(x)$ and $f_n(y)$ differ only at the $i(n)$-th bit. More precisely, f is distinction-intractable at the $i(n)$-th bit if for each probabilistic polynomial time algorithm M, for each polynomial Q, for all sufficiently large n, $\Pr\{f_n(x) \neq_{i(n)} f_n(y)\} < 1/Q(n)$, where $(x, y) = M(f)$ and $x_1 \neq_{i(n)} x_2$ means that x_1 and x_2 differ only at the $i(n)$-th bit. It is not hard to verify that distinction-intractableness implies one-wayness.

Theorem 3 *Assume that f is a permutation that is distinction-intractable at the $i(n)$-th bit. Then the hash function H constructed in Section 4.2 is a CIH.*

Proof : Assume for contradiction that H is not a CIH. Then there are a polynomial Q, an infinite subset $\mathbf{N}' \subseteq \mathbf{N}$ and a probabilistic polynomial time algorithm M such that M on input $h \in_R H_n$ finds with probability $1/Q(n)$ a collision-pair (x, x'), for all $n \in \mathbf{N}'$.

Since $h(x) = h(x')$, there is an $1 \leq m \leq \ell(n) - 1$ such that $\mathrm{ins}_{i(n)}(y_{m-1}, x_{\ell(n)-m+1}) \neq \mathrm{ins}_{i(n)}(y'_{m-1}, x'_{\ell(n)-m+1})$ and $\mathrm{ins}_{i(n)}(y_{j-1}, x_{\ell(n)-j+1}) = \mathrm{ins}_{i(n)}(y'_{j-1}, x'_{\ell(n)-j+1})$ for each $m < j \leq \ell(n)$. Here x_i, x'_i, y_i and y'_i are defined in the same way as in the proof for Theorem 2. It is not hard to see that $f_n(\mathrm{ins}_{i(n)}(y_{m-1}, x_{\ell(n)-m+1}))$ and $f_n(\mathrm{ins}_{i(n)}(y'_{m-1}, x'_{\ell(n)-m+1}))$ differ only at the $i(n)$-th bit, i.e., $f_n(\mathrm{ins}_{i(n)}(y_{m-1}, x_{\ell(n)-m+1})) \neq_{i(n)} f_n(\mathrm{ins}_{i(n)}(y'_{m-1}, x'_{\ell(n)-m+1}))$. Thus for each $n \in \mathbf{N}'$, M can be used to find, with the same probability $1/Q(n)$, a pair of strings $w(= \mathrm{ins}_{i(n)}(y_{m-1}, x_{\ell(n)-m+1}))$ and $w'(= \mathrm{ins}_{i(n)}(y'_{m-1}, x'_{\ell(n)-m+1}))$ such that $f_n(w) \neq_{i(n)} f_n(w')$. This contradicts the assumption that f is distinction-intractable at the $i(n)$-th bit, and the theorem follows. $\square$

In [Dam89] a CIH is constructed under the assumption of the existence of *claw-free pairs of permutations*. Let f^0 and f^1 be permutations over $\bigcup_n \Sigma^n$. Intuitively, (f^0, f^1) is a *claw-free pair of permutations* if for all sufficiently large n, it is computationally infeasible to find a pair of strings (x, y) such that $x, y \in \Sigma^n$, $x \neq y$ and $f^0_n(x) = f^1_n(y)$. From a claw-free pair of permutations (f^0, f^1), one constructs a function $h : \bigcup_n \Sigma^{n+1} \to \bigcup_n \Sigma^n$ as follows: For each n, let $h_n(x' \diamond x_1) = f^{x_1}_n(x')$, where $x' \in \Sigma^n$ and $x_1 \in \Sigma$. Let h_n be an instance of H_n, and let $H = \bigcup_n H_n$. In [Dam89] H was proved to be a CIH.

Now we show a relationship between distinction-intractable permutations and claw-free pairs of permutations:

Theorem 4 *Assume that f is a permutation that is distinction-intractable at the $i(n)$-th bit. Let f' be the permutation defined by $f'_n(x) = f_n(x) \oplus 0^{n-i(n)}10^{i(n)-1}$, where $x \in \Sigma^n$. Then (f, f') is a claw-free pair of permutations.*

Proof : Assume that we can find two strings $x, y \in \Sigma^n$ such that $x \neq y$ and $f(x) = f'(y)$. Then $f(x) = f(y) \oplus 0^{n-i(n)}10^{i(n)-1}$, i.e. $f(x) \neq_{i(n)} f(y)$. This is a contradiction. $\square$

It is not clear whether or not the inverse of the above theorem is also true, i.e., whether or not we can construct a distinction-intractable permutation from a claw-free pair of permutations.

6 Practical One-Way Hash Functions Are Easy to Find

In this section we show that ideas underlying Theorems 2 and 3 can be used to design *practical* one-way hash functions. The fastest of these hash functions compress nearly $2n$-bit long input into n-bit long output strings by applying only *twice* a one-way function.

Let $f : D \to R$ be a one-way function where $D = \bigcup_n \Sigma^n$, $R = \bigcup_n \Sigma^{m(n)}$ and m is a polynomial. Let k be a real with $k > 1$, s an integer with $s \geq 2$ and $\ell(n) = s(n - \lfloor n/k \rfloor)$. Typically, we choose $1 < k \leq 10$ and $2 \leq s \leq 5$. For each $\alpha \in \Sigma^{\lfloor n/k \rfloor}$, associate with it a function h_α defined by: $y_0 = \alpha, y_1 = \mathrm{tail}_{\lfloor n/k \rfloor}(f_n(y_0 \diamond x^s)), \ldots y_j = \mathrm{tail}_{\lfloor n/k \rfloor}(f_n(y_{j-1} \diamond x^{s-j+1})), \ldots h_\alpha(x) = f_n(y_{s-1} \diamond x^1)$. where $x = x^s \cdots x^2 x^1 \in \Sigma^{\ell(n)}$ and $x^1, x^2, \ldots, x^s \in \Sigma^{n - \lfloor n/k \rfloor}$. Let $H_n = \{h_\alpha | \alpha \in \Sigma^{\lfloor n/k \rfloor}\}$ and $H = \bigcup_n H_n$.

In practice, we first choose, uniformly at random, a string α from $\Sigma^{\lfloor n/k \rfloor}$, and fix it. Then by using the function h_α as is defined above, we can compress $\ell(n)$-bit input into n-bit output strings. This procedure is called the *Hashing Method*, and can be used as a basic step of the serial compressing method defined in Lemma 4 and the parallel compressing method mentioned in Section 3.2.

We were not able to prove that the hash function H is a CIH or a UOH/$EN[\ell]$, even under the assumption that 1 up to $n - \lfloor n/k \rfloor$ are all simultaneously hard bits of f. A sufficient condition for H to be a CIH is that *it is computationally difficult to find two distinct strings* $x, y \in \Sigma^n$ *such that* $\mathrm{tail}_{\lfloor n/k \rfloor}(f_n(x)) = \mathrm{tail}_{\lfloor n/k \rfloor}(f_n(y))$. A (secure) public-key encryption function can be viewed as a function satisfying the condition. For a common-key block cipher, the function from its key space to ciphertext space induced by a randomly chosen plaintext can also be viewed as a function satisfying the condition.

It seems that, when f is carefully chosen, H is *strong* enough and also *efficient* enough for practical applications. In the remaining portion of this section, we present two concrete examples. One is based on the Rabin encryption function, and the other on a common-key block cipher called xDES. There is another good example based on the RSA encryption function with low exponents. Discussions for it are analogous to the first example, and hence omitted here.

6.1 Compressing via the Rabin Encryption Function

Let $M_n = pq$ where p and q are $n/2$-bit long randomly generated primes. Denote by Z_{M_n} the residue classes of integers modulo M_n. The Rabin encryption function *rabin* is defined by $rabin_n(x) = x^2 \bmod M_n$ where $x \in Z_{M_n}$. For Blum integers M_n, i.e., $p \equiv q \equiv 3 \pmod 4$, it was proved that 1 up to $O(\log n)$ are simultaneously hard bits of *rabin*. Now let $k = 10$. When n is large (say ≥ 500), as the authors know, no currently existing algorithms can efficiently find two distinct elements $x, y \in Z_{M_n}$ such that the last $n/10$ bits of $x^2 \bmod M_n$ coincide with that of $y^2 \bmod M_n$.

By the use of the Rabin encryption function and the Hashing Method, we can compress 900-bit input to 500-bit output strings by performing only *twice* the multiplication of two 500-bit integers modulo an integer of the same length. This procedure can be implemented very efficiently, even by software.

6.2 Compressing via xDES

Let m be a polynomial. Informally, a common-key block cipher with length $m(n)$ is a pair of polynomial time computable functions $(encrypt, decrypt)$, where $encrypt$ and $decrypt$ are functions from $\bigcup_n \Sigma^n \times \Sigma^{m(n)}$ to $\Sigma^{m(n)}$ that have the following properties:

1. $ptxt = decrypt_n(key, encrypt_n(key, ptxt))$ for all $key \in \Sigma^n$ and all $ptxt \in \Sigma^{m(n)}$.
2. It is computationally difficult to find $ptxt$ from $encrypt_n(key, ptxt)$ for any $ptxt \in \Sigma^{m(n)}$, without knowing key.

Let f be the function from $\bigcup_n \Sigma^n$ to $\bigcup_n \Sigma^{m(n)}$ that is defined by $f_n(x) = encrypt_n(x, \alpha)$, where $\alpha \in_R \Sigma^{m(n)}$. Then each f_n can be used to compress strings by the Hashing Method. To prevent the compressing method from *meet-in-the-middle attacks*, n should be chosen in such a way that $m(n)$ is sufficiently large, say > 120. A rigorous treatment of this subject can be found in [NS90].

Consider the perhaps most widely used modern encryption algorithm DES. According to our definition, DES is the restriction of *some* common-key block cipher on $\Sigma^{56} \times \Sigma^{64}$. DES should not be directly used to compress strings by the Hashing Method, for its key length is too short. Now we use DES as bricks to build a common-key block cipher called xDES. Our building method is based on a theory on the construction of secure block ciphers developed in [ZMI89].

Let r be a polynomial with $r(i) \geq 2i+1$. xDES is defined by xDES^0, xDES^1, xDES^2, xDES^3, $\cdots$, where xDES^0 is the same as DES and, for each $i \geq 1$, xDES^i is a function from $\Sigma^{56r(i)i} \times \Sigma^{128i}$ to Σ^{128i} consisting of $r(i)$ rounds of *Type-2 transformations* [ZMI89]. More details follow.

1. The definition for xDES^0: Same as DES.
2. The definition for xDES^i where $i \geq 1$: Let $key = key_{r(i),i} \diamond \cdots \diamond key_{r(i),2} \diamond key_{r(i),1} \diamond \cdots \diamond key_{2,i} \diamond \cdots \diamond key_{2,2} \diamond key_{2,1} \diamond key_{1,i} \diamond \cdots \diamond key_{1,2} \diamond key_{1,1}$ and $ptxt = ptxt_{2i} \diamond \cdots \diamond ptxt_2 \diamond ptxt_1$, where $key_{i_1,i_2} \in \Sigma^{56}$ and $ptxt_{i_3} \in \Sigma^{64}$ for all $1 \leq i_1 \leq r(i)$, $1 \leq i_2 \leq i$ and $1 \leq i_3 \leq 2i$. Then $ctxt = \mathrm{xDES}^i(key, ptxt)$ is computed as follows:

   Step 0: Let $c_{0,1} = ptxt_1, c_{0,2} = ptxt_2, \cdots, c_{0,2i-1} = ptxt_{2i-1}, c_{0,2i} = ptxt_{2i}$.

   Step j, for each $1 \leq j \leq r(i)$: Let $c_{j,1} = c_{j-1,2i}, c_{j,2} = c_{j-1,1} \oplus \mathrm{DES}(key_{j,1}, c_{j-1,2}), \cdots, c_{j,2i-1} = c_{j-1,2i-2}, c_{j,2i} = c_{j-1,2i-1} \oplus \mathrm{DES}(key_{j,i}, c_{j-1,2i})$.

   Step $r(i)+1$: Let $ctxt = c_{r(i),2i} \diamond \cdots \diamond c_{r(i),2} \diamond c_{r(i),1}$.

Let $r(i) = 2i+1$ and $k = 3$. Using the Hashing Method, we can compress 224-bit input into 128-bit output strings by performing only twice xDES^1, i.e., 6 times DES.

Finally, we note that xDES can also be used in normal encryption/decryption operations, and DES can be replaced by any other secure common-key block encryption algorithm.

References

[BM84] M. Blum and S. Micali, How to generate cryptographically strong sequences of pseudo-random bits, SIAM J. on Comp. 13 (1984) 850-864.

[BH89] R. Boppana and R. Hirschfeld, Pseudorandom generations and complexity classes, in: S. Micali, ed., Randomness and Computation, (JAI Press Inc., 1989) 1-26.

[Dam89] I. Damgård, A design principle for hash functions, Presented at Crypto'89 (1989).

[GGM86] O. Goldreich, S. Goldwasser and S. Micali, How to construct random functions, J. of ACM 33 (1986) 792-807.

[GM84] S. Goldwasser and S. Micali, Probabilistic encryption, J. of Comp. and Sys. Sci. 28 (1984) 270-299.

[ILL89] R. Impagliazzo, L. Levin and M. Luby, Pseudo-random generation from one-way functions, Proc. of the 21-th ACM STOC (1989) 12-24.

[IL89] R. Impagliazzo and M. Luby, One-way functions are essential for complexity based cryptography, Proc. of the 30-th IEEE FOCS (1989) 230-235.

[Mer89] R. Merkle, One way hash functions and DES, Presented at Crypto'89 (1989).

[MSc88] S. Micali and C.P. Schnorr, Super-efficient, perfect random number generators, in: S. Goldwasser, ed., Proc. of Crypto'88, (Springer-Verlag, 1990) 173-198.

[NY89] M. Naor and M. Yung, Universal one-way hash functions and their cryptographic applications, Proc. of the 21-th ACM STOC (1989) 33-43.

[NS90] K. Nishimura and M. Sibuya, Probability to meet in the middle, J. of Cryptology 2 (1990) 13-22.

[WC81] M. Wegman and J. Carter, New hash functions and their use in authentication and set equality, J. of Comp. and Sys. Sci. 22 (1981) 265-279.

[Yao82] A. Yao, Theory and applications of trapdoor functions, Proc. of the 23-th IEEE FOCS (1982) 80-91.

[ZMI89] Y. Zheng, T. Matsumoto and H. Imai, On the construction of block ciphers provably secure and not relying on any unproved hypotheses, Presented at Crypto'89, (1989).

[ZMI90a] Y. Zheng, T. Matsumoto and H. Imai, Connections among several versions of one-way hash functions, Proc. of IEICE of Japan E73 (July 1990).

[ZMI90b] Y. Zheng, T. Matsumoto and H. Imai, Structural properties of one-way hash functions, Presented at Crypto'90, (1990).

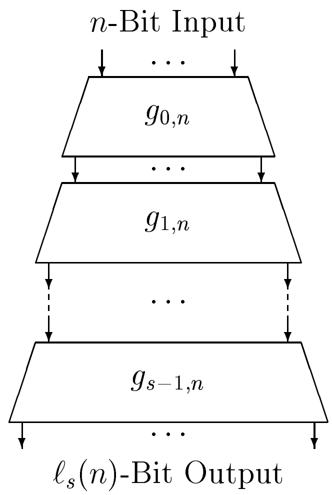

Figure 1: Serial-Extending 1

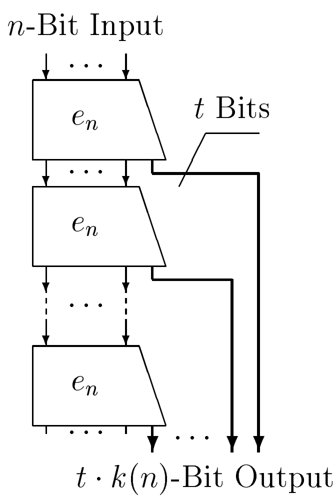

Figure 2: Serial-Extending 2

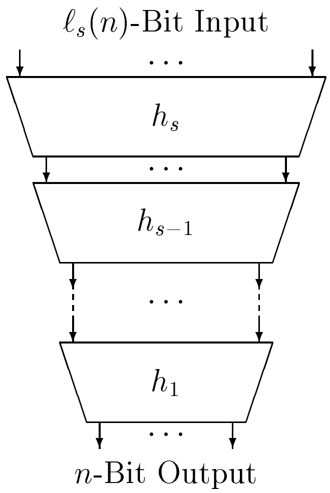

Figure 3: Serial-Compressing 1

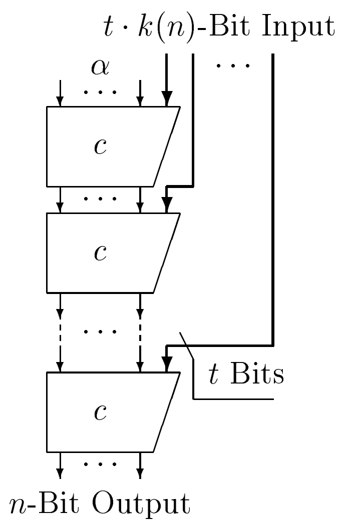

Figure 4: Serial-Compressing 2

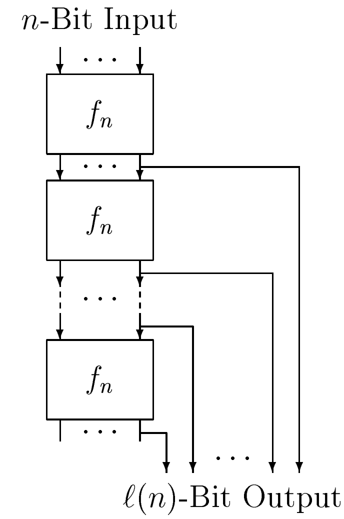

Figure 5: Construction of PSG

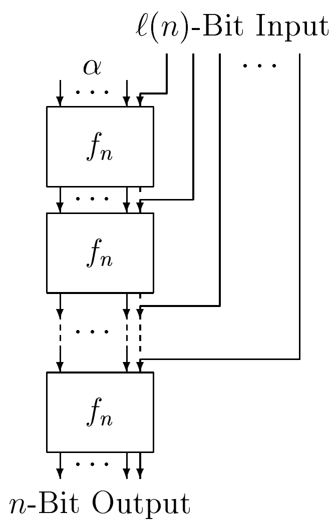

Figure 6: Construction of UOH/$EN[\ell]$