

# Cryptanalysis-tolerant Hashing and Commitment

**Extended Abstract (abridged) version:** Friday, January 24, 2003

-- Comments Welcome!! --

Amir Herzberg, [amir@herzberg.name](mailto:amir@herzberg.name), <http://amir.herzberg.name>

Computer Science Dept.

Bar-Ilan University

## *Abstract*

*Cryptographic mechanisms should be **cryptanalysis-tolerant**, i.e., i.e. secure as long as one of multiple cryptographic assumptions (functions) resist cryptanalysis efforts. We present provably cryptanalysis-tolerant compositions of several cryptographic mechanisms. In particular, we present cryptanalysis-tolerant design for keyless hash functions with strong confidentiality, randomness-extraction and collision-resistance properties. The compositions or the hash functions can be used for cryptanalysis-tolerant commitment scheme. Our compositions are efficient and practical; they may be used with existing cryptographic functions or to design new functions. Our definitions of properties of keyless hash functions may be of independent interest.*

**Keywords:** cryptographic functions; hash functions; one-way functions; collision-resistance; collision-resistant hash functions; commitment schemes; pseudo-randomness; randomness extraction; extractors; t-resilient; exposure-resilient functions, min-entropy; MD5; SHA; RIPE-MD; TLS.

## 1. Introduction

Most cryptographic functions do not have an unconditional proof of security. The classical method to establish security is by *cryptanalysis*, i.e. accumulated evidence of failure of experts to find weaknesses in the function. However, cryptanalysis is an expensive, time-consuming and fallible process. In particular, since a seemingly-minor change in a cryptographic function may allow an attack which was previously impossible, cryptanalysis allows only validation of specific functions and development of engineering principles, and does not provide a solid theory for designing cryptographic functions. Indeed, it is impossible to predict the rate or impact of future cryptanalysis efforts; a mechanism which was attacked unsuccessfully for years may abruptly be broken by a new attack<sup>1</sup>. Hence, it is desirable to design systems to be *cryptanalysis tolerant*, namely so the system remains secure following successful cryptanalysis of one or few cryptographic mechanisms. Cryptanalysis tolerance does not imply unconditional-security, since security still depends on the resistance of cryptographic mechanisms to cryptanalysis. However, cryptanalysis tolerance would hopefully provide sufficient time to replace the broken cryptographic functions.

One way to ensure cryptanalysis-tolerance is to use provable constructions of cryptographic mechanisms from few `basic` cryptographic mechanisms, which have simple cryptanalysis-tolerant designs. For example, many cryptographic mechanisms can be constructed from one-way functions; and trivially it is sufficient that one of  $\{g, g'\}$  is a one-way function, to ensure their composition  $f(x, x') = g(x) || g'(x')$  is also a one-way function. Provably-secure constructions based on one-way functions exist for many cryptographic mechanisms, e.g. pseudo-random generators [Go01, HILL99] and signature schemes [NY89]. Therefore, by using as the one-way function a cryptanalysis-tolerant composition of multiple functions assumed to be one-way, the mechanisms retain the proven security properties even if one of the functions is not one-way. However, such constructions are often inefficient, and often also involve unacceptable degradation in security parameters (e.g., require absurd key and/or block sizes); see quantitative / concrete security analysis [HL92, BKR94].

In this paper we focus on an alternative, *direct approach* for achieving (proven) cryptanalysis tolerance. Consider a design of a cryptographic function  $f$  satisfying some goal  $\Pi$ , using  $m > 1$  cryptographic functions  $f_1, \dots, f_m$ . The design is *cryptanalysis-tolerant for  $\Pi$*  if when  $m-1$  of the

---

<sup>1</sup> In practice, we try to use conservative estimates of progress in cryptanalysis, based on past progress and other factors.

functions satisfy  $\Pi$ , then  $f$  also satisfies  $\Pi$ . The design *preserves*  $\Pi$  if when all  $m$  functions satisfy  $\Pi$ , then  $f$  also satisfies  $\Pi$ . Otherwise, the design *does not preserve*  $\Pi$ . Cryptanalysis-tolerance may be applied in the internal design of cryptographic functions, or by composing several functions.

We are not aware of cryptanalysis-tolerance identified explicitly as a general goal (or given a term) in previous works. However, there are several known, simple compositions, widely used in applied cryptography for cryptanalysis-tolerance. We present most of these designs, and their impact for several important cryptographic mechanisms, in Table 1. In particular, the *parallel composition*,  $g(x)||g'(x)$ , using the same input  $x$  to both functions, ensures cryptanalysis-tolerance for several *integrity* properties, such as (several variants of) collision-resistant hashing as well as Message Authentication Codes (MAC) and digital signatures. The parallel composition is used in practical designs and standards, e.g. in the TLS protocol [RFC2246] and the XML-DSIG specifications [add citation]. The TLS standard contains additional cryptanalysis-tolerant compositions, to ensure security as long as at least one of two standard hash functions (MD5 and SHA-1) satisfies certain security properties. This improves on the (older) SSL protocol, which *also* combined MD5 and SHA-1, but clearly failed to ensure cryptanalysis-tolerance; see details of both TLS and SSL in [R00].

Another simple and well known composition provides cryptanalysis-tolerance for several confidentiality properties (although not for one-way functions). This is the *cascade* or *sequential composition*, in which a number of cryptographic functions are applied sequentially. In particular, an ancient cryptographic technique is *cascaded encryption*, where a message  $m$  is encrypted using  $E_1$  and  $E_2$  by applying them sequentially, i.e.

$E[k_1||k_2](m)=E_1[k_1](E[k_2](m))$ . The related problem of cascade of ciphers (pseudo-random permutations) was analyzed by [ABCV98, EG85, MM93]. Cascading is used in the internal design of most practical ciphers as well as in sensitive applications.

However, there are many important cryptographic mechanisms for which there is no known cryptanalysis-tolerant design. In particular, this holds for several important cryptographic primitives that combine confidentiality and integrity properties; these primitives are often critical for important applications and protocols. Some examples include:

- General-purpose, standard cryptographic hash functions such as MD5 [R92], SHA-1 [FIP180] and RIPEMD-160 [PBD97] (assumed to be, among other things, collision resistant *and* one-way).
- Rigor definitions of categories or properties of hash functions combining confidentiality and integrity, e.g. Perfectly One Way (POW) hashing<sup>2</sup> [C97, CMR98].
- Commitment schemes see e.g. [DPP94, DPP98, HM96].

Cryptanalysis-tolerance is especially important for hash functions and commitment schemes, which are widely used in practice, but almost always using keyless designs, while known provably-secure constructions are keyed. Furthermore, we believe less cryptanalysis efforts were directed at hashing compared to ciphers and cryptosystems.

We present cryptanalysis-tolerant compositions for keyed and keyless cryptographic hash functions, satisfying multiple security properties, including confidentiality, collision-resistance and randomness extraction. These are the first provably-secure cryptanalysis-tolerant compositions of general cryptographic functions, beyond the simple, classical compositions in Table 1. Few additional cryptographic constructions were proven secure based on validity of either of two (specific) `hardness` assumptions, see e.g. [Sh00, O92].

**Organization.** In Section 2 we briefly recap definitions of standard cryptographic concepts and primitives. In Section 3 we present simple compositions of standard cryptographic functions, such as cascade and parallel compositions. In Section 4 we present the  $E$ -composition, a combination of cascade and parallel compositions, that ensures cryptanalysis-tolerance for multiple properties of hash functions, including collision-resistance and confidentiality properties. In Section 5 we present new definitions of properties of keyless cryptographic functions; these definitions are based on assuming minimal amount  $m$  of

---

<sup>2</sup> Perfectly one-way hashing was referred to as Oracle hashing in [C97], and renamed as POW in [CMR98].

entropy to the input to the function ( $m$ -min-entropy). In Section 6 we present the  $EZ$ -composition, extending the  $E$ -composition to provide cryptanalysis-tolerance for  $m$ -min-entropy cryptographic hash functions. We conclude with open questions in Section 7.

## 2. Standard Definitions

We now present few conventions used in this paper, as well as standard definitions of cryptographic mechanisms; these definitions are included only for completeness and when significantly used in this paper. Definitions which are new to this paper are collected in the last subsection of this section.

For simplicity, in the current version of the paper we adopt the asymptotic, polynomial-time definitions; namely, we assume that the adversary has the probabilistic polynomial time (PPT) computational abilities. We believe that our results translate well to concrete definitions and analysis as in [BR97, BDJR97], and plan to extend them accordingly in the final version of this paper. We adopted several definitions and notations from [CMR98].

### Notations.

We usually use  $n$  for the length of the input and  $k$  for the length of the output. For  $y \in \{0, 1\}^n$ , let  $y[i]$  denote the  $i^{\text{th}}$  bit of  $y$ , and  $y[i \dots j]$  denote the substring of  $y$  consisting of the  $i^{\text{th}}$  to the  $j^{\text{th}}$  bit. Therefore,  $y = y[1 \dots n]$ .

Let  $p$  be a permutation over  $\{1 \dots n\}$ ; define  $p(y) \in \{0, 1\}^n$  as  $p(y)[i] = y[p(i)]$ .

We use capital letters to denote random variables, sets and probability distributions. Let  $\text{Pr}\{X=x\}$  denote the probability that random variable  $X$  will assume the value  $x$ . Let  $x \in_R X$  denote the choice of  $x$  according to distribution  $X$ , or uniformly from set  $X$ . Let  $U_n$  denote the uniform distribution over  $\{0, 1\}^n$ .

For a given random variable  $X$  over domain  $D$ , denote by  $\|X\|$  the *infinity norm* of  $X$ , which is the maximal probability for any element  $d \in D$ ; namely  $\|X\| = \max\{\text{Pr}(X=d) : d \in D\}$ . The *min-entropy* of  $X$ , denoted  $H_\infty(X)$ , is  $H_\infty(X) = -\log\|X\|$ . Namely, if the min-entropy of  $X$  is  $k = H_\infty(X)$ , then the probability of any element in  $D$  is bounded by  $2^{-k}$ .

### 2.1. Basic Cryptographic Definitions

A function  $f(n)$  is **negligible** (up to a polynomial) if for every polynomial  $p$  there exists an integer  $N$  s.t. for all  $n > N$  holds  $|f(n)| < 1/p(n)$ . We denote  $f(n) \approx_p 0$ . Two functions  $f, g$  for which  $|f(n) - g(n)| \approx_p 0$  are said to **have polynomially-negligible difference**, and we write  $f \approx_p g$ .

A **probability ensemble** is a sequence  $\{X_n\}$  of random variables  $X_n$  over  $\{0, 1\}^n$ . Two probability ensembles  $\{X_n\}, \{Y_n\}$  are **computationally indistinguishable** if for all probabilistic polynomial time predicates  $A$  holds  $\text{Pr}\{A(X_n)\} \approx_p \text{Pr}\{A(Y_n)\}$ ; we denote  $\{X_n\} \approx_p \{Y_n\}$ .

Most of our results concern cryptographic functions which use at least one key, often a public key known (also) to the adversary. In this work, we focus on functions computable by uniform PPT algorithms, whose keys and inputs chosen as arbitrary random bit strings; see [G01] for more general definitions. Since we use asymptotic, poly-time definitions, we consider arbitrary input length  $n$  and corresponding key length  $K(n)$  and output length  $\lambda(n)$ .

We say that  $f$  is a **keyed poly-time computable function with key length  $K(n)$  and output length  $\lambda(n)$** , or simply that  $f$  is a **keyed function**, if for every  $x \in \{0, 1\}^n$  and every  $k \in \{0, 1\}^{K(n)}$ , there is a PPT algorithm that computes  $f_k(x) \in \{0, 1\}^{\lambda(n)}$ . If  $\lambda(n) < n$ , then  $f$  is a **keyed hash function**, and if  $\lambda(n) = n$  then  $f$  is **keyed length-preserving function**. If  $f$  is a keyed length-preserving function and  $f_k$  is a permutation for all  $n$  and all  $k \in \{0, 1\}^{K(n)}$ , then  $f$  is a **keyed permutation**.

### 2.2. Confidentiality and Randomness properties

Let  $f$  be a keyed function. Assume that for every PPT predicate  $A^g$  with oracle access to a function  $g: \{0, 1\}^n \rightarrow \{0, 1\}^{\lambda(n)}$  holds  $\text{Pr}_{k \in U_{K(n)}}\{A^{f_k}(I^n)\} \approx_p \text{Pr}_{g \in G^{(n)}}\{A^g(I^n)\}$ , where  $G^{(n)}$  is the set of all functions from  $\{0, 1\}^n$  to  $\{0, 1\}^{\lambda(n)}$ . Then  $f$  is a **pseudo-random function (PRF)**. If

$f_k$  is a regular function for all  $k$ , and  $G^{(n)}$  is the set of all regular functions from  $\{0,1\}^n$  to  $\{0,1\}^{\lambda(n)}$ , then  $f$  is a **pseudo-random regular function (PRRF)**. Similarly if  $\lambda(n)=n$  and  $G^{(n)}$  is the set of all permutations over  $\{0,1\}^n$ , then  $f$  is a **pseudo-random permutation (PRP)**. We say that  $f$  is a **computationally-secure extractor for length  $n$  with min-entropy  $l$** , if for  $k \in_R U_n$  and a random variable  $X$  with min-entropy at least  $l$ , the result of  $f_k(X) \approx_p U_{\lambda(n)}$ . An extractor is called **strong** if also  $\langle k, f_k(X) \rangle \approx_p \langle U_{k(n)}, U_{\lambda(n)} \rangle$ . Most of the work on extractors require the output distribution to be statistically-close to uniform (rather than computationally-indistinguishable as we do); our results apply also to the (stronger) statistically-close definition. We preferred to use the weaker notion since it could be assumed (and validated by cryptanalysis) for 'practical' hash functions.

**ADD:** definitions of one-way functions (OWF), one-way regular functions (OWRF), one-way permutations (OWP), and encryption schemes see literature, e.g. [BJDR97].

ERF (define and add also chosen??)

"Random oracle"

### 2.3. Collision-resistance and Integrity properties

We say that  $f$  is **(any) collision resistant (ACR)** if for any PPT algorithm  $A$ , the probability of  $A$  finding a collision is negligible, i.e.  $\Pr\{A(k)=(x;x'):x \neq x', f_k(x)=f_k(x')\} \approx_p 0$ , where the probability is taken over the keys  $k \in U_{k(n)}$ . We say that  $f$  is **target collision resistant (TCR)** if for any PPT algorithm  $A$  and string  $x$ , the probability of  $A$  finding a collision  $x'$  to  $x$  is negligible, i.e.  $\Pr\{A(k)=x':x \neq x', f_k(x)=f_k(x')\} \approx_p 0$ , where the probability is taken over the keys  $k \in U_{K(n)}$ . If  $f$  is also a keyed hash function, then we say that  $f$  is **Any (Target) collision resistant hash function**, or simply that  $f$  is **Any-CRHF** (respectively, **Target-CRHF**).

A keyless function  $h$  is **weakly collision resistant hash function (weakly CRHF)** if for sufficiently long inputs,  $|h(x)| < |x|$ , and for any PPT algorithm  $A$ , the probability of  $A$  finding a collision to a randomly-selected input  $x$  is negligible, i.e.  $\Pr\{A(x)=x':x \neq x', h(x)=h(x')\} \approx_p 0$ , where the probability is taken over the input  $x \in_R U_n$ .

Add definitions for: *MAC, Signature*

### 2.4. Commitment: Requiring Confidentiality and Collision resistance

Several cryptographic mechanisms combine confidentiality properties with collision-resistance properties. Possibly the most widely used primitive with both properties is a commitment scheme; we focus on non-interactive commitment schemes. Commitment schemes receive an input message  $x$ , and produce two outputs: a commitment  $c$  and a decommitment  $d$ , such that  $c$  and  $d$  together reveal  $x$ . The confidentiality property is that  $c$  by itself does not expose  $m$ . A collision is a commitment  $c$  together with two pairs:  $\langle x, d \rangle$  and  $\langle x', d' \rangle$ , such that  $\langle c, d \rangle$  reveals  $x$  while  $\langle c, d' \rangle$  reveals  $x' \neq x$ .

Like collision-resistant hash functions, rigor studies focus on keyed commitment schemes, to avoid adversaries which have 'wired in' collisions. A **non-interactive commitment scheme** consists of three keyed functions: *commit*, *decommit*, and *reveal*, all of which with key length  $K(n)$  and with two input parameters (in addition to the key). The inputs to the *commit* and *decommit* functions are a message  $x$  of length  $n$ , and a random string  $r \in_R U_{R(n)}$ , and it holds that:  $x = \text{reveal}_k(\text{commit}_k(x, r), \text{decommit}_k(x, r))$ .

A commitment scheme is **hiding** if  $\text{commit}_k(x, r)$  exposes no information about  $x$ , i.e. for any two messages  $x, x'$  holds:  $\langle k, \text{commit}_k(x, r) \rangle \approx_p \langle k, \text{commit}_k(x', r) \rangle$ , for  $k \in_R U_{k(n)}$  and  $r \in_R U_{R(n)}$ . A commitment scheme is **binding** if for any PPT algorithm  $A$ , the probability of  $A$  finding a collision is negligible, i.e.  $\Pr\{A(k)=(c; \langle x, d \rangle; \langle x', d' \rangle): x \neq x', x = \text{reveal}_k(c, d), x' = \text{reveal}_k(c, d')\} \approx_p 0$ , for  $k \in_R U_{k(n)}$  and  $r \in_R U_{R(n)}$ . A commitment scheme is **secure** if it is hiding and binding.

## 3. Basic Compositions

In this section we explore several few basic, 'classical' cryptanalysis-tolerant compositions of two cryptographic mechanisms. In this version of the paper we define the compositions only

for cryptographic properties of two keyless functions  $f, g$  or two keyed functions  $f_k, g_k$ . When composing keyed algorithms, we use different, independently-chosen keys for the two algorithms. Most compositions have natural extensions for more complex mechanisms, such as.

We describe and define each composition in the following subsections. In Table 1, we list the results of using each of the compositions for the `standard` cryptographic properties of keyed and keyless functions presented in the previous section. The table also contains results of corresponding compositions for encryption, signature and commitment schemes, each being a set of related functions (e.g. encryption and decryption) rather than a single function. The following lemma states the properties summarized in the table.

**Lemma 1 (Basic compositions of standard primitives):** Let  $T[p, c]$  denote the entry of Table 1 at row with property  $p$  and column with composition  $c$ . Then:

**If  $T[p, c] = \text{Tolerant}$**  then composition  $c$  is cryptanalysis-tolerant for property  $p$ ; namely the composition  $c[f, g]$  has the property  $p$ , if either  $f$  or  $g$  has the property  $p$ .

**If  $T[p, c] = \text{Ok}$**  if the composition  $c[f, g]$  preserves property  $p$ ; namely it does not ensure cryptanalysis-tolerance, but on the other hand, if both  $f$  and  $g$  has the property  $p$ , then their composition  $c[f, g]$  will also have the property  $p$ .

**If  $T[p, c] = \text{X}$**  if there is a counterexample where  $f$  and  $g$  have the property  $p$ , but their composition  $c[f, g]$  does not have property  $p$ .

**Otherwise, i.e.  $T[p, c] = ?$**  if we do not know whether property  $p$  is preserved under composition  $c$ .

### 3.1. Cascade Composition

The most basic cryptanalysis-tolerant composition of cryptographic functions is by cascade. Indeed it is very natural, even without precise analysis, to believe that the cascade of two cryptosystems  $E, E'$  is at least as secure as the more secure of the two, and it seems reasonable to hope that it is even more secure than both. Indeed, cascading of cryptosystems has been a common practice in cryptography for hundreds of years.

The *cascade* of keyless algorithms  $f, g$  is denoted  $f \circ g$  and defined as  $f \circ g(x) = f(g(x))$ . The *cascade* of keyed algorithms  $f_k, g_k$  is denoted  $f_k \circ g_k$  and defined as  $f_k \circ g_k(x) = f_k(g_k(x))$ .

### 3.2. Parallel (Same-Input Multiple-Outputs) Composition

We now consider parallel compositions of cryptographic functions. We begin with a very useful composition: parallel application of two cryptographic functions to the same input, where the output is the concatenation of the outputs of both functions. We call this the *Same-Input, Multiple-Outputs (SIMO) Parallel Composition*, or simply the *parallel composition*. The *parallel composition* of  $f, g$  is denoted as  $f || g$ , and defined as  $f || g(x) = f(x) || g(x)$ . The *parallel composition* of two keyed functions  $f_k, g_k$  is denoted  $f_k || g_k$  or simply as  $f || g$ , and defined as  $f_k || g_k(x) = f_k(x) || g_k(x)$ .

The parallel composition provides cryptanalysis-tolerance for most `integrity cryptographic primitives` such as collision-resistant hash functions, MAC and signature schemes. The parallel composition preserves most confidentiality primitives, such as OWF, PRF, and encryption schemes.

### 3.3. Same-Input XOR-Output (XOR) Composition

The output of the parallel composition is the concatenation of the outputs of the two functions; therefore if one of the two functions does not has weak privacy or randomness properties, the composition will also be weak in the corresponding property. For some applications, we do not need both outputs, and we can combine them to protect the secrecy of the input and improve the randomness of the output. A simple and natural way to combine the outputs is using XOR; this is the *Same-Input, XOR-Output Parallel Composition*, or simply the *XOR composition*. We assume that the functions involved have the same (fixed) output length, to allow bitwise XOR of the outputs.

The XOR composition of  $f, g$  is denoted  $f \oplus g$  and defined as  $f \oplus g(x) = f(x) \oplus g(x)$ , where  $\oplus$  denotes bit-wise exclusive OR. The XOR composition of  $f_k, g_k$  is denoted  $f_k \oplus g_k$  (or simply by  $f \oplus g$ ), and defined as  $f_k \oplus g_k(x) = f_k(x) \oplus g_k(x)$ .

The XOR composition is applied in the TLS standard, to combine two pseudo-random functions (one based on MD5 and the other based on SHA-1). This provides cryptanalysis-tolerance.

A variant of the XOR composition is used in the internal design of the RIPE-MD cryptographic hash function. The compression function in RIPE-MD,  $RIPEmd$ , is the composition of two compression functions,  $c_L$  and  $c_R$ , with output of 160 bits each, as follows:  $RIPEmd(x) = c_L + x[1 \dots 160] + c_R$ . The  $+$  operator views each of its 160-bit operands as five words of 32 bits each; its result is also five words of 32 bits, where each word is the result of addition modulo  $2^{32}$  of the corresponding words from the two operands. Except for using this  $+$  operand rather than XOR, this is the same as the XOR composition (for the two keyless compression functions  $c_L$  and  $c_R$ ).

So far we did not find security properties for keyless functions, where the XOR composition (or the variant in RIPE-MD) provides cryptanalysis-tolerance. For example, if  $f=g$  (for XOR) or  $f$  is chosen such that  $f(x)+g(x)=0$  (for RIPE-MD), then finding (second) pre-images becomes trivial.

Furthermore, the XOR composition is bad also for (keyed) collision-resistant hash functions. For example, let  $\{f_k: \{0,1\}^* \rightarrow \{0,1\}^L\}$  be an ACR hash function family.

Define  $g_k(x) = \begin{cases} 0z & \text{for } x = 0^*z, \text{ where } z \in \{0,1\}^L \\ f_k(x) & \text{Otherwise} \end{cases}$ ; clearly  $g_k$  is also ACR but the

XOR composition of two instances of  $g_k$  (with independently chosen keys) is obviously not ACR (or even TCR).

Practitioners may protest that this example is artificial, since in practice  $f$  and  $g$  would be different and 'independent' (the functions used in RIPE-MD are indeed different, although they share many details to simplify implementation). Indeed, if we restrict ourselves to analysis in the 'random oracle model', we find that the XOR composition does provide cryptanalysis-tolerance. Of course, this is not provable security, which should be preferred when feasible.

**Claim:** Assume protocol  $\Pi$  is secure when function  $h$  is implemented by a random oracle. Then  $\Pi$  is secure also when  $h = f \oplus g$ , where either  $f$  or  $g$  is implemented by a random oracle.

### 3.4. Multiple-Inputs XOR-Output (MIXOR) Composition

Assume  $f$  provides pseudo-random output given inputs from a certain distribution; the XOR composition of  $f$  may yet be easily distinguishable from random. For example, by composing  $f$  with itself (i.e. with  $g=f$ ), we get  $f(x) \oplus f(x) = 0$ . One way to fix this is by computing each of the functions  $f, g$  on different parts of the input. We call this the *Multiple-Inputs XOR-Output (MIXOR) Parallel Composition*, or simply the *MIXOR composition*. We assume both functions have the same (fixed) input length  $n$  and output length  $l$ . The composition has double input length  $2n$ , and the same output length  $l$ .

The MIXOR composition of  $f, g$  is denoted  $\oplus[f||g]$  and defined as

$\oplus[f||g](x) = f(x[1 \dots n]) \oplus g(x[n+1 \dots 2n])$ , where  $\oplus$  denotes bit-wise exclusive OR. The XOR composition of  $f_k, g_k$  is denoted  $\oplus[f_k||g_k]$  and defined as  $\oplus[f_k||g_k](x) = f_k(x[1 \dots n]) \oplus g_k(x[n+1 \dots 2n])$ .

### 3.5. Multiple-Inputs Multiple-Outputs (MIMO) Composition

The parallel composition applied both cryptographic functions to the same input. We can simplify it further and apply each of the two functions to a different part of the input string; this is the *Multiple-Inputs, Multiple-Outputs (SIMO) Parallel Composition*, or simply the

*MIMO composition.* We focus on MIMO compositions of keyless functions with fixed input length  $n$  and output length  $l$ . The composition has double input and output lengths. The *MIMO composition* of  $f, g: \{0, 1\}^n \rightarrow \{0, 1\}^l$  maps  $x \in \{0, 1\}^{2n}$  to  $f(x[1 \dots n]) \parallel g(x[n+1 \dots 2n]) \in \{0, 1\}^{2l}$ . The *MIMO composition* of  $f_k, g_k$  is  $f_k(x[1 \dots n]) \parallel g_k(x[n+1 \dots 2n])$ .

### 4. The *E* Composition: Cryptanalysis-Tolerant Confidentiality and Integrity

In the previous sections we saw simple compositions (cascade, parallel) ensuring cryptanalysis-tolerance for integrity (collision resistance), randomness extraction or confidentiality properties. However, many applications require more than one of these properties. In particular, multiple properties are required to develop standard, 'multipurpose' hash functions. In particular, none of the simple compositions seems to work well for functions with both confidentiality and collision resistance properties. In fact, we did not (yet) find a composition ensuring both confidentiality and collision-resistance properties using only *two* candidate/component functions. However, we found a composition that provides cryptanalysis-tolerance for several properties, by composing *three* candidate functions  $\{h_0, h_1, h_2\}$  (two of which must have each desired property). This is the *E*-composition, which is illustrated in Figure 1, for composing keyed as well as keyless functions, and defined as follows:

For keyless functions, the *E*-composition is:

$$E[h_1, h_2, h_3](x) = h_0(h_1(x)) \parallel h_1(h_2(x)) \parallel h_2(h_0(x)).$$

For keyed functions: similar, except that *E* also has a key,  $k = k[i, j]$  for  $i = 0, 1, 2$  and  $j = 1, 2$ , which is split to each of the functions. Namely, the *E*-composition is:

$$E[h_1, h_2, h_3]_k(x) = h_{0, k[0, 2]}(h_{1, k[1, 1]}(x)) \parallel h_{1, k[1, 2]}(h_{2, k[2, 1]}(x)) \parallel h_{2, k[2, 2]}(h_{0, k[0, 1]}(x)).$$

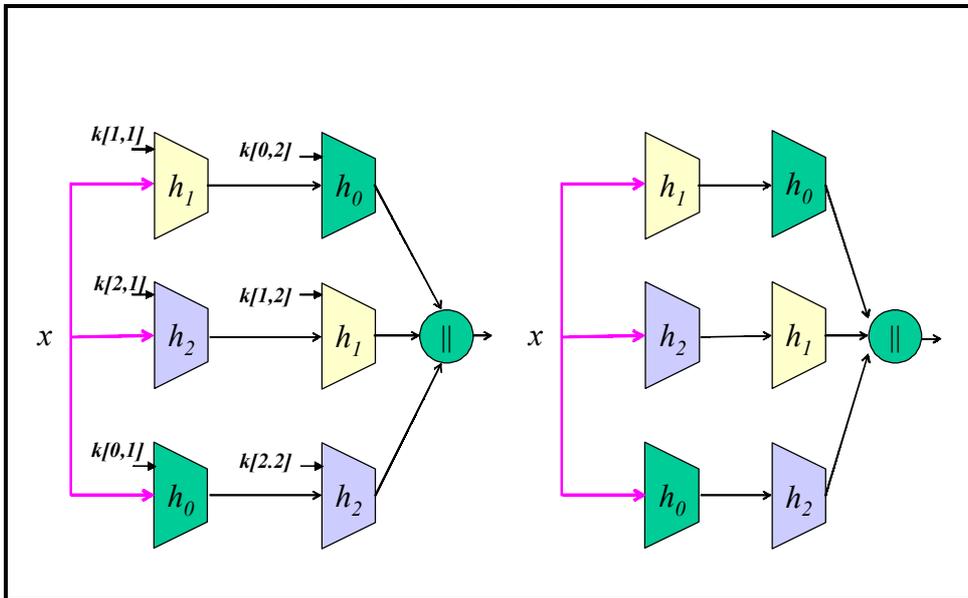


Figure 1: The *E*-Composition for keyed (left) and keyless (right) functions

The *E*-composition combines the cascade and parallel compositions: in fact, it is the parallel composition of three cascades, each of them of one of the three possible pairs of two of the three candidate functions. It is based on the following observation (which can be generalized for other combinations and/or compositions):

**Lemma 2:** Let  $p$  be a property of keyed or keyless functions, which is *preserved under both* cascade and parallel composition, and *cryptanalysis-tolerant under either* cascade or parallel composition. Then  $p$  is cryptanalysis-tolerant under the *E*-composition.

The only property of keyless functions which satisfies the conditions in Lemma 2 is the One-Way Regular Function (OWRF) property. This, by itself, is not very useful, as we could simply use the cascade composition to achieve cryptanalysis-tolerant composition of OWRFs. In the next section we present alternative confidentiality, collision-resistance and randomness properties for keyless functions; and in Section 6 we present the *EZ*-composition, providing cryptanalysis-tolerance for these properties.

We get some more results by applying Lemma 2 to keyed functions. It follows immediately from Table 1 and Lemma 1 that:

**Lemma 3:** The *E*-composition ensures cryptanalysis-tolerance for the PRRF, Any CRHF, Target CRHF, and MAC properties of keyed functions.

Most of these are integrity properties, which are tolerant also under the simpler parallel composition; PRRF is the only exception (a confidentiality property, tolerant under cascade). Therefore, we can use the *E*-composition to ensure cryptanalysis-tolerance to functions which are both collision-resistant or MAC, and PRRF.

#### 4.1. *E*-composition of commitment schemes

A more interesting application of the *E*-composition is for commitment schemes. We define the *E*-composition of non-interactive commitment schemes. Such schemes consist of three keyed functions: *commit*, *decommit*, and *reveal*. Both *commit* and *decommit* are probabilistic, i.e. have random input in addition to the message. The *reveal* function is deterministic, but it also has two input parameters – the results of the commit and the decommit functions:  $x = \text{reveal}_k(\text{commit}_k(x,r), \text{decommit}_k(x,r))$ . The multiple functions in the scheme and the multiple parameters make the definition more hairy, but the principle is the same as of the *E*-composition of keyed functions as in Figure 1.

The *E*-composition for three commitment schemes  $C_i = \langle c_i, d_i, r_i \rangle$ , for  $i=0,1,2$ , is the triplet  $E[C_1, C_2, C_3] = \langle C, D, R \rangle$  defined as follows (see illustration of  $C, D$  in Figure 2). Each of the functions  $\langle C, D, R \rangle$  has a key  $k$ , which consists of six sub-keys:  $k = k[i,j]$  for  $i=0,1,2$  and  $j=1,2$ . Similarly, functions  $C$  and  $D$  have a random input string  $r$ , which also consists of six sub-strings:  $r = r[i,j]$  for  $i=0,1,2$  and  $j=1,2$ . Each of the functions is defined as follows:

$$C_k(x,r) = c_{0,k[0,2]}(c_{1,k[1,1]}(x,r[1,1]),r[0,2]) \parallel c_{1,k[1,2]}(c_{2,k[2,1]}(x,r[2,1]),r[1,2]) \parallel c_{2,k[2,2]}(c_{0,k[0,1]}(x,r[0,1]),r[2,2]).$$

$$D_k(x,r) = d_{0,k[0,2]}(c_{1,k[1,1]}(x,r[1,1]),r[0,2]) \parallel d_{1,k[1,1]}(x,r[1,1]) \parallel$$

$$d_{1,k[1,2]}(c_{2,k[2,1]}(x,r[2,1]),r[1,2]) \parallel d_{2,k[2,1]}(x,r[2,1]) \parallel$$

$$d_{2,k[2,2]}(c_{0,k[0,1]}(x,r[0,1]),r[2,2]) \parallel d_{0,k[0,1]}(x,r[0,1])$$

Let the input to  $R_k$  be  $c = \langle c_1, c_2, c_0 \rangle$  and  $d = \langle d_{1,1}, d_{2,1}, d_{0,1}, d_{1,2}, d_{2,2}, d_{0,2} \rangle$ . For  $i=0,1,2$ , let  $j = i-1 \bmod 3$ . Then  $x_i = r_{i,k[i,1]}(r_{j,k[j,2]}(c_{j,2}, d_{j,2}), d_{i,1})$ . If  $x_0 = x_1 = x_2$ , then  $R_k(c,d) = x_0$ ; otherwise,  $R_k(c,d)$  returns error indication.

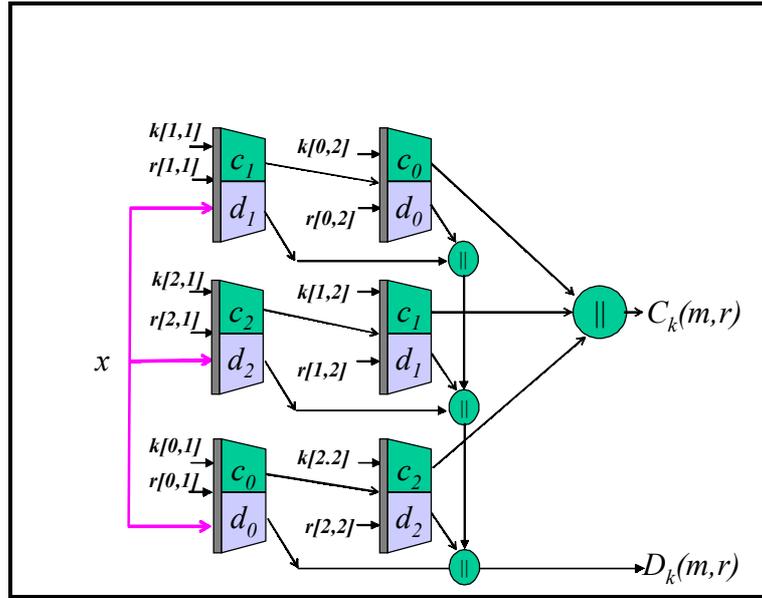


Figure 2: The  $E$ -Composition for non-interactive, keyed commitment schemes

It follows from Table 1 and Lemma 1 that:

**Lemma 4:** The  $E$ -composition ensures cryptanalysis-tolerance for both binding and hiding properties of non-interactive commitment schemes.

Therefore, the  $E$ -composition provides a cryptanalysis-tolerant design for secure (binding and hiding) non-interactive, keyed commitment schemes.

## 5. Min-Entropy-Properties for Keyless Functions

All standard hash functions (SHA, MD5, RIPEMD, etc.) are keyless, however most of the rigor work so far focused on keyed functions (or ensembles). We believe it is important to investigate formally keyless cryptographic functions. As a first step, we now define strong properties for keyless hash functions. Additional work will be required to refine these definitions and verify whether they are meaningful and feasible. We focused on properties which compose well, to ensure cryptanalysis-tolerance.

### 5.1. Definitions of Min-Entropy-Based Properties

We begin with the randomness property. Exposure-resilient functions already provide a strong randomness property for keyless functions. However, we did not find any cryptanalysis-tolerant composition for exposure-resilient functions; the existence of such compositions remains an open question.

We define a different randomness property for keyless functions: *min-entropy preserving*. Intuitively, we require that if the input to the function has at least some minimal entropy, then the output of the function will also have this entropy. By itself, this property is trivial to achieve, e.g. by using  $h(x)=x$ . However when combined with other properties, the min-entropy preserving property becomes meaningful.

An efficiently-computable function  $h$  is  **$m$ -min-entropy extracting** if for every efficiently-generated input distribution  $X$  s.t.  $H_\infty(X) > m$ , holds  $H_\infty(h(X)) > m$ . Suppose that the range of  $h$  is  $\{0, 1\}^{2k}$ , with  $k > m$ . If  $H_\infty(h(X)[1 \dots k]) > m$  and  $H_\infty(h(X)[k+1 \dots 2k]) > m$ , then  $h$  is **double  $m$ -min-entropy extracting**.

The double min-entropy preserving property allows us to use only half of the bits of  $h$ , which is very useful. It can be extended to other subsets of the output of  $h$ , however for our purposes the simple `double` property suffices.

Notice that exposure-resilient functions ensure that the output is indistinguishable from uniform, while the output of min-entropy extracting functions may be easily distinguishable from uniform, e.g. the least significant bit may always be zero. On the other hand, exposure-resilient functions require that the input is uniformly random. Min-entropy extracting

functions relax this by allowing non-uniform input (as long as it has min-entropy at least  $m$ ). This allows the adversary to choose<sup>3</sup> most of the input bits.

The min-entropy extracting property is cryptanalysis-tolerant under the parallel composition, and preserved under cascade, as stated in the following lemmas.

**Lemma 5:** Given  $f, g \rightarrow \{0,1\}^k$ , one of which is (double)  $m$ -min-entropy extracting, then  $h(x)=f(x)||g(x)$  is (double)  $m$ -min-entropy extracting.

**Lemma 6:** Given two (double)  $m$ -min-entropy extracting functions  $f, g$ , such that the range of  $g$  is in the domain of  $f$ . Then  $h(x)=f(g(x))$  is (double)  $m$ -min-entropy extracting hash function. We now define a min-entropy based confidentiality, one-way property for keyless functions. Our definition extends the usual definition of one-way functions, by allowing the adversary to specify the input distribution, as long as it has min-entropy of at least  $m$ .

An efficiently-computable function  $h$  is  **$m$ -min-entropy one-way** if for every efficiently-generated input distribution  $X$  s.t.  $H_\infty(X) > m$ , and for every PPT adversary  $A$ , holds:

$$\text{Prob}[h(A(h(x)))=h(x)] \approx_p 0.$$

The  $m$ -min-entropy one-way property seems substantially stronger than the `regular` one-way function property.

We now define a min-entropy based collision-resistance (integrity) property for keyless functions. Like for weak collision-resistance, the adversary will need to find a collision to a randomly-selected input. However, like for  $m$ -min-entropy one-way, we allow the adversary to select any input distribution with min-entropy at least  $m$ . In particular the adversary can chose an input distribution where part of the input is fixed. If we allowed the adversary to choose the entire input, this becomes `strong collision resistance`, which is impossible to achieve for keyless hash functions.

We say that hash function  $h: \{0,1\}^* \rightarrow \{0,1\}^k$  is  **$m$ -min-entropy Collision Resistant** if for every efficiently-generated input distributions  $X$  s.t.  $H_\infty(X) > m$  and every PPT adversary  $A$ , holds  $\text{Pr}\{A(x)=x' \text{ s.t. } x \neq x' \text{ and } h(x)=h(x')\} \approx_p 0$ , where the probability is taken over  $x \in_R X$ .

Sometimes, an application requires resistance not to `regular` collisions but to other variant; see e.g. in [A93]. In particular, we define resistance to `half collisions`, i.e. inputs  $x \neq x'$  s.t.  $h(x)[1 \dots k/2] = h(x')[1 \dots k/2]$  or  $h(x)[k/2 \dots k] = h(x')[k/2 \dots k]$ . If the definition holds also for half-collisions, we say that  $h$  is  **$m$ -min-entropy Half-Collision Resistant**.

In many cases, an application requires multiple properties. An  **$m$ -min-entropy Cryptographic Hash**, or simply a **cryptographic hash function**, is a function  $h: \{0,1\}^* \rightarrow \{0,1\}^k$  which is double  $m$ -min-entropy extracting,  $m$ -min-entropy hiding and  $m$ -min-entropy Half-Collision Resistant. Cryptographic hash functions are useful for many purposes. In particular, the trivial construction of commitment scheme from hash function, widely-adopted by practitioners [Sc96], is  $C_k(x,r)=h(x,k,r)$ ,  $D_k(x,r)=(x,r)$ ,  $R_k(c,d)=\{x \text{ if } d=(x,r) \text{ and } c=h(x,k,r), \text{ otherwise error indication}\}$ . We observe that:

**Lemma 7:** Let  $h$  be an  $m$ -min-entropy Cryptographic Hash function. Let, Then  $C=\langle C,D,R \rangle$  is a secure (binding and hiding) commitment scheme.

Therefore, we can achieve a cryptanalysis-tolerant commitment scheme either by using the  $E$ -composition of candidate commitment schemes, as in Lemma 4, or by using the trivial construction above, using a cryptanalysis-tolerant  $m$ -min-entropy Cryptographic Hash function. The latter design seems to offer several advantages, in particular, reduced amount of keying and randomization material. Indeed, we present a cryptanalysis-tolerant composition for cryptographic hash functions in the next section. We now explain why we cannot simply use the  $E$ -composition for this.

In Table 2 we summarize the results of composing functions with different  $m$ -min-entropy properties, using the cascade, parallel and XOR compositions. Notice that the `multiple-input`

<sup>3</sup> In  $t$ -resilient functions [CG\*85], which are the same as perfect exposure resilient functions, the adversary is also allowed to choose (up to  $t$ ) of the input bits.

compositions (MI-XOR, MIMO) do not work well for these properties, since each part of the input may contain less than  $m$  entropy<sup>4</sup>.

Unfortunately, we find that neither the one-way nor the collision-resistant properties persist under both the cascade and parallel compositions. Therefore, we cannot apply the  $E$ -composition, using Lemma 2, to provide cryptanalysis-tolerant min-entropy cryptographic hash (with confidentiality and collision-resistance properties). In the next section we present the  $EZ$ -composition, an extension of the  $E$ -composition which provides cryptanalysis-tolerance for min-entropy based properties of keyless functions.

## 6. The $EZ$ -Composition

In Figure 3 we present the  $EZ$ -composition<sup>5</sup> of three keyless functions,  $\{h_0, h_1, h_2\}$ . Let  $x_i = h_i(x)$  and  $H_i = h_i(x_{i+1 \bmod 3} || x_{i-1 \bmod 3} [1 \dots L/2])$ . Then  $EZ[h_1, h_2, h_3](x) = H_0 || H_1 || H_2$ .

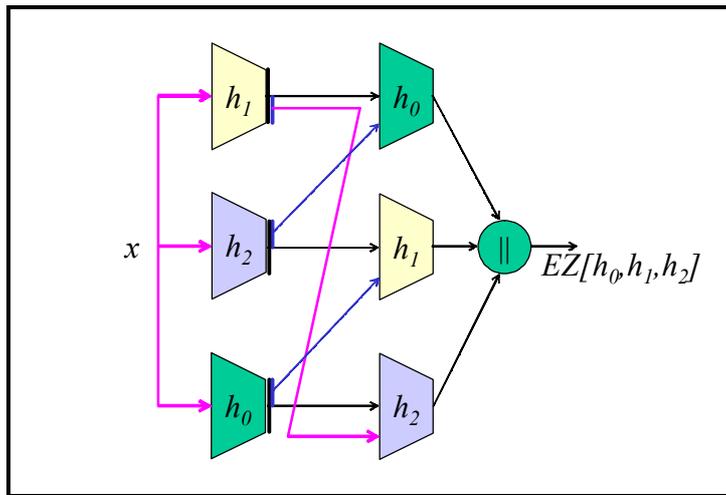


Figure 3: The  $EZ$ -Composition for Keyless Functions

The min-entropy is preserved in the intermediate values  $x_i$  and  $x_i[1 \dots L/2]$ .

**Lemma 8:** If  $h_i$  is  $m$ -min-entropy double extracting, and  $x \in_{\mathcal{R}X}$  s.t.  $H_\infty(X) > m$ , then the min-entropy of  $m_i$  and  $m_i[1 \dots L/2]$  is at least  $m$ .

We obtain our main result:

**Theorem 1** The  $EZ$ -composition ensures cryptanalysis-tolerance for cryptographic hash functions.

## 7. Conclusions and Open Questions

In this work we presented new definitions for keyless cryptographic hash functions, which appear useful for many applications, as well as cryptanalysis-tolerant compositions. We believe these definitions allow rigorous research of practical hash functions and their applications. There are many directions for further research based on the current work, including:

1. Further research of keyless cryptographic functions. In particular, is there a cryptanalysis-tolerant composition for Exposure-resilient functions? Are there provably-secure constructions of min-entropy cryptographic hash functions, from established cryptographic primitives, such as pseudo-random function families and one-way functions?
2. Can we define variants of the  $E$  and/or  $EZ$  compositions which will provide cryptanalysis-tolerance for stronger definitions of commitment schemes, in

<sup>4</sup> An extension of the definitions, e.g. requiring min-entropy from each half of the input, may allow multiple-input compositions.

<sup>5</sup> The name  $EZ$  was chosen since graphical representation of the  $EZ$ -composition contains the letters  $E$  and  $Z$  (in purple).



- [Protection Against Break-ins](#), RSA CryptoBytes, Volume 3, No. 1, 1997.
- [CG\*85] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolensky, "The Bit Extraction Problem or t-Resilient Functions," Proc. 26 th IEEE Symposium on Foundations of Computer Science, 1985, pages 396--407.
- [CMR98] R. Canetti, D. Micciancio and O. Reingold, "Perfectly One-Way Probabilistic Hash Functions," Proceedings of 30th STOC, 1998.
- [Da89] Ivan B. Damgård. A design principle for hash functions. Advances in Cryptology – proc. of Crypto '89, Lecture Notes in Computer Science, Volume 435, Springer-Verlag, New York, 1990, pp. 416-427.
- [DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In Proceedings of the 23rd Symposium on Theory of Computing, ACM STOC, 1991.
- [DF89] Y. Desmedt and Y. Frankel. *Threshold cryptosystems*. In Advances in Cryptology--Crypto '89, pages 307--315, 1989.
- [Do98] Hans Dobbertin, Cryptanalysis of MD4, Journal of Cryptology, Volume 11, Number 4, 1998.
- [Do00] Yevgeniy Dodis. Exposure-Resilient Cryptography. Ph.D. Thesis., MIT, 2000.
- [DPP94] Ivan B. Damgård, Torben P. Pedersen, Birgit Pfitzmann: On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures; Crypto '93, LNCS 773, Springer-Verlag, Berlin 1994, 250-265.
- [DPP98] Ivan B. Damgård, Torben P. Pedersen, Birgit Pfitzmann: Statistical Secrecy and Multi-Bit Commitments; IEEE Transactions on Information Theory 44/3 (1998) 1143-1151.
- [EG85] S. Even and O. Goldreich, On the Power of Cascade Ciphers, ACM Transactions on Computer Systems, Vol. 3, 1985, pp. 108-116.
- [FIP180] National Institute of Standards and Technology, Federal Information Processing Standards Publication, FIPS Pub 180-1: Secure Hash Standard (SHA-1), April 17, (1995), 14 pages.
- [Go01] Oded Goldreich, The Foundations of Cryptography, Volume 1 (Basic Tools), ISBN 0-521-79172-3, Cambridge University Press, June 2001.
- [Go02] Oded Goldreich, Fragments of a Chapter on Encryptions Schemes, Extracts from working drafts of Volume 2, The Foundations of Cryptography.
- [GIL\*90] O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesen, D. Zuckerman. "Security preserving amplification of randomness", 31st Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, (1990), 318-326.
- [GGM84] Oded Goldreich and Shafi Goldwasser and Silvio Micali "How to Construct Random Functions" Journal of the ACM, 33(4), 1984, 792-807.
- [H02b] Amir Herzberg, Introduction to Secure Communication and Commerce using Cryptography, Chapter 3 – Cryptographic functions without secret key, draft of book to be published by Prentice-Hall, available online at <http://amir.herzberg.name/book.html>.
- [HILL99] Johan Hastad, Rudich Impagliazzo, Leonid A. Levin, and Mike Luby, Construction of a Pseudorandom Generator from any One-Way Function. SIAM Journal on Computing, Vol. 28, No. 4, pp. 1364-1396, 1999.
- [HL82] Amir Herzberg and Mike Luby, "Public Randomness in Cryptography", proceedings of CRYPTO 1992, ICSI technical report TR-92-068, October, 1992.
- [HM96] S. Halevi and S. Micali, "Practical and Provably-Secure Commitment Schemes from Collision Free Hashing", in Advances in Cryptology - CRYPTO96, Lecture Notes in Computer Science 1109, Springer-Verlag, 1996, pp. 201-215.
- [Me89] R.C. Merkle. One way hash functions and DES. Advances in Cryptology – proc. of Crypto '89, Lecture Notes in Computer Science, Volume 435, Springer-Verlag, New York, 1990, pp. 428—446; based on Ph.D. thesis, Stanford, 1979.
- [MM93] U.M. Maurer and J.L. Massey, Cascade ciphers: the importance of being first, Journal of Cryptology, Vol. 6, No. 1, pp. 55-61, 1993.
- [MOV96] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, [Handbook of Applied Cryptography](#), Section 9.2.6, CRC Press, ISBN 0-8493-8523-7, October 1996. Available online at <http://www.cacr.math.uwaterloo.ca/hac/>.
- [NN99] Noam Nisan and Amnon Ta-Shma. Extracting randomness: A survey and new constructions. Journal of Computer and System Sciences, 58(1):148--173, 1999.
- [NY89] Moni Naor, Moti Yung: Universal One-way Hash Functions and their Cryptographic Applications; 21<sup>st</sup> Symposium on Theory of Computing (STOC), 1989, ACM, New York, pp. 33-43.
- [O92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, Advances in Cryptology--CRYPTO '92, volume 740 of Lecture Notes in Computer Science, pages 31--53. Springer-Verlag, 1992.
- [PBD97] Bart Preneel, Antoon Bosselaers, and Hans Dobbertin. The cryptographic hash function RIPEMD-160. RSA Laboratories CryptoBytes newsletter, 3(2):9-14, Autumn, 1997. Also see A. Bosselaers, H. Dobbertin, B. Preneel, "The RIPEMD-160 cryptographic hash function," Dr. Dobb's Journal, Vol. 22, No. 1, January 1997, pp. 24--28.
- [RFC2246] T. Dierks, C. Allen, The TLS Protocol: Version 1.0, Network Working Group, Internet Engineering Task Force (IETF). Available online at <http://www.ietf.org/rfc/rfc2246.txt>.
- [R92] Ronald Rivest, "The MD5 message digest algorithm," Internet RFC 1321, 1992, <http://theory.lcs.mit.edu/rivest/Rivest-MD5.txt>.
- [R00] Eric Rescorla. SSL and TLS: Designing and Building Secure Systems. Addison-Wesley, 2000.
- [RSW00] Omer Reingold, Ronen Shaltiel, and Avi Wigderson. Extracting randomness via repeated condensing. In Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, pages 22--31, 2000.

- [Sc96] Bruce Schneier, *Applied Cryptography*, John Wiley & Sons, 1996.
- [Sh79] Adi Shamir, How to Share a Secret, *Communications of the ACM* 22, 1979, pp. 612--613.
- [Sh00] Victor Shoup, *Using hash functions as a hedge against chosen ciphertext attacks*, *Adv. in Cryptology -- Proc. of Eurocrypt '2000*, LNCS 1807, pp. 275-288.
- [T92] Gene Tsudik, "Message authentication with one-way hash functions", *Proceedings of Infocom*, 1992.
- [TV00] L. Trevisan, S. Vadhan, *Extracting randomness from samplable distributions*, pp. 32-42, 41st Annual Symposium on Foundations of Computer Science (FOCS), November 12 - 14, 2000, Redondo Beach, California.

Design:	Cascade	Parallel	XOR	MI-XOR	MIMO
	 $f(g(x))$	 $f(x)  g(x)$	 $f(x) \oplus g(x)$	 $f(x_1) \oplus g(x_2)$	 $f(x_1)  g(x_2)$
OWF	X	X	X	?	Tolerant
OWRF	Tolerant	Ok	X	?	Tolerant
OWP	Tolerant	X	X	X	X
PRF	Ok	Ok	Tolerant	Tolerant	X
PRRF	Tolerant	Ok	Tolerant	Tolerant	X
PRP	Tolerant <sup>6</sup>	?	Tolerant	X	X
Encryption	Tolerant	Ok	X	X	Ok
Weak CRHF	X	Tolerant	X	X	Ok
Any CRHF	Ok	Tolerant	X	X	Ok
TargetCRHF	Ok	Tolerant	X	X	Ok
MAC	Ok	Tolerant	X	X	Ok
Signature	Ok	Tolerant	X	X	Ok
ERF, PRG	Ok	X	X	Tolerant	Ok
(Strong) Extractor	Ok	X	Tolerant (add $k$ bits)	Tolerant	Ok
“Random Oracle”	Ok	Ok	Tolerant	Ok	Ok
Commit. scheme Hiding	Tolerant	Ok	Tolerant	Ok	Ok
Commit. scheme Binding	Ok	Tolerant	X	X	X

Table 1: Cascade and Parallel Composition of ‘Standard’ Cryptographic Primitives

Design:	Cascade	Parallel	XOR
---------	---------	----------	-----

<sup>6</sup> The cascade of two permutation ensembles is a pseudo-random permutation ensemble, provided (at least) one of the two is a pseudo-random permutation ensemble. Of course, cascading a PRP with a function which is not a permutation will not result in a PRP.

Schematic:			
Output	$f(g(x))$	$f(x)  g(x)$	$f(x)\oplus g(x)$
Extracting	Ok	Tolerant	X
One-way	X	X	X
Hiding and Extracting	Ok	X	X
Collision-Resistant	X	Tolerant	X
Crypto Hash (all properties)	X	X	X

**Table 2: Cascade and Parallel Composition of  $m$ -Min-Entropy Properties**