Cryptanalysis-tolerant Commitment and Hashing

Draft version: Thursday, August 29, 2002

-- Comments Welcome!! –

Amir Herzberg, amir@herzberg.name
Computer Science Dept.
Bar-Ilan University

Abstract

Cryptographic solutions should be cryptanalysis-tolerant, i.e. avoid dependency on the assumed security of a single cryptographic function. We present the **2/3 composition***, a cryptanalysis-tolerant design for commitment schemes and cryptographic hash functions. Previous cryptanalysis-tolerant solutions provided either confidentiality or binding properties; the 2/3 composition provides both properties. The 2/3 composition is simple and efficient, and appropriate for practical applications, either to compose existing functions or to design new functions.*

Keywords: cryptographic functions; hash functions; one-way functions; collision-resistance; commitment schemes

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, rather than provide a theory for designing cryptographic functions. Indeed, it 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. 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 would hopefully provide sufficient time to replace the broken cryptographic functions.

There are two basic approaches to achieving (proven) cryptanalysis tolerance. The first approach is to construct complex cryptographic mechanisms from simpler, cryptanalysis-tolerant functions. For example, many works construct different cryptographic mechanisms, e.g. pseudo-random generators [Go01,HILL99] and signature schemes [NY89], from one-way functions. One-way functions have a trivial cryptanalysis-tolerant composition, namely the *parallel composition* $f(x_1||x_2)=f_1(x_1)||f_2(x_2)$ is a one-way function as long as *either* f_1 or f_2 is a one-way function. Therefore, by using a composition of multiple functions assumed to be one-way, the mechanisms retain the proven security properties even if one of the functions is not a

one-way function. 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]. Only relatively few constructions are sufficiently simple and efficient, while preserving concrete security, to be of practical use. Many of these transform a cryptographic function with limited domain, typically *Fixed Input Length (FIL)*, into a function (meeting the same goal) with extended domain, typically *Variable Input Length (VIL).* The design and cryptoanalytical verification of FIL functions is much easier than that of VIL functions, therefore these constructions are very useful in practice. Important extensions from FIL to VIL include *Cipher Block Chaining (CBC)* [BKR94,BDJR97], extending pseudo-random permutations (ciphers) and MAC functions, and the *Merkle-Damgård cascade*, extending pseudo-random functions [BCK96F] and often used to extend collision-resistant hash functions [Da89,Me89,BR97].

In this paper we focus on the *other approach* for achieving (proven) cryptanalysis tolerance, namely *direct cryptanalysis-tolerant design by redundancy.* Redundancy is a basic engineering goal for reliable and secure systems, ensuring tolerance to failures, corruptions and key exposures; see e.g. [CGHN97,DF89,Sh79]. For cryptanalysis-tolerance, we design cryptographic function f satisfying some goal Π, using two or more cryptographic functions $f_1,...f_m$, such that if at least t out of the m functions satisfy Π, then f also satisfies Π. Redundancy should be applied in the internal design of cryptographic functions, or by applying the composition of 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 at least two known, simple compositions, widely used in applied cryptography for cryptanalysis-tolerance. The first is the *cascade* or *sequential composition,* in which a number of cryptographic functions are applied sequentially. In particular, the oldest composition of cryptographic function 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], and in fact cascading is widely used in the internal design of ciphers as well as in their application. Cascading provides cryptanalysis-tolerance for some *confidentiality properties*, such as ciphers and one-way permutations, and appropriate definitions of encryption schemes; however, not for all confidentiality functions, e.g. not for one-way functions.

The other simple and widely used composition for cryptanalysis tolerance is the *parallel composition.* We mentioned that the parallel composition of two candidate one-way functions $f_1(x_1)$ and $f_2(x_2)$ with independent inputs x_1 and x_2, namely $f_1(x_1)||f_2(x_2)$, is a one-way function as long as either f_1 or f_2 is a one-way function; however, this is not really a good composition, as it requires a longer block. Clearly, the related *parallel input composition*, i.e. $f(x)= f_1(x)||f_2(x)$, is *not* cryptanalysis-tolerant; in fact even if f_1 and f_2 are one-way functions, $f_1(x)||f_2(x)$ may not be a one-way function.

However, the *parallel input composition* provides good cryptanalysis-tolerance for some *integrity properties.* For example, a collision for $f(x)=f_1(x)||f_2(x)$ implies a collision for both f_1 and f_2. Indeed the parallel compositions of collision-resistant hash

functions, Message Authentication Codes (MAC) and digital signatures are all cryptanalysis-tolerant (while the cascade of these functions is clearly *not* cryptanalysis-tolerant).

However, several important cryptographic primitives 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 hash functions combining confidentiality and integrity properties, e.g. Oracle Hashing [C97], Perfectly One Way (POW) hashing [CMR98].
- Commitment schemes see e.g. [DPP94, DPP98, HM96].

Unfortunately, the `classical` cascade and parallel compositions do not seem to apply when *both* confidentiality and integrity properties are involved. The reader may find it useful to construct counter-examples – it is quite easy. We will pick as an example the internal design of standard hash function. These functions all use the Merkle-Damgård cascade to transform a FIL hash function, usually called a *compression function*, into a VIL hash function. It is easy to see that if the compression function is one-way and strongly collision-resistant, than so is hash function obtained by the Merkle-Damgård construction. However, let us consider now the construction of the compression function itself, focusing on MD5 for example. As illustrated in Figure 1, the MD5 compression function has two inputs: the current block to be compressed *x[i]*, and the `state` *s[i]*. The function applies, sequentially, four `candidate compression functions` $c_1, \ldots c_4$ to the current block, where each application uses the state as the result from the previous application.

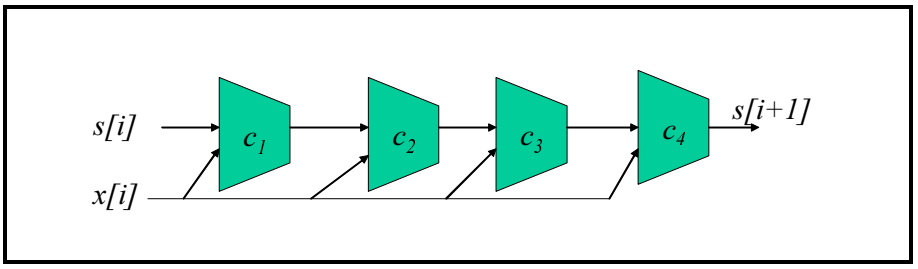

Figure 1: Design of the Compression Function of MD5

Users of such compositions may hope that the composition ensures cryptanalysis-tolerance. In particular, they may hope that it suffices for one of the $c_1, \ldots c_4$ to be one-way and collision resistant function, for the composite compression function to be one-way and collision resistant. We caution that for an arbitrary choice of $c_1, \ldots c_4$, this hope may not be justified. As a trivial example, if c_4 outputs the same value regardless of its input, then clearly the composite function is neither collision-resistant nor one-way – regardless of $c_1, \ldots c_3$. It is easy to come up with less trivial examples as well.

Of course, such examples do not imply that the specific compression functions are not secure; they merely point out that the composition does not ensure cryptanalysis-tolerance. In particular, as long as the entire compression function is not cryptanalyzed successfully, one may use it in the hope that it is secure enough,

similarly to the naïve usage of encryption schemes (actually, since ciphers are cryptanalysis-tolerant, users can feel safer relying on the fact that only few rounds of the cipher were broken). However we caution that it is not clear how much cryptanalysis efforts were directed at such functions, and therefore the limited number of published attacks may not necessarily mean that the schemes are sufficiently secure for a particular application. Notice in particular that published attacks seem to have used very limited resources and were published by relatively few cryptanalysts (see e.g. [Do98]).

Our contributions. We provide the first cryptanalysis-tolerant compositions for cryptographic primitives providing both confidentiality and integrity. The main primitive is a new form of a hash function we define and call *commit-hash function.* A commit-hash function satisfies several properties required from `practical` cryptographic hash functions, such as collision resistance and one-way (input hiding). Furthermore, using commit-hash functions, we can prove the security of the `folklore` commitment scheme (commit by sending *h(m,r)* using `standard` crypto-hash *h* with random *r,* open by sending *m,r*).

We show a simple composition of three commit-hash functions, called the **2/3 composition**, whose security does not depend on the least-secure of the three functions (i.e., is about as secure as the second-best function); therefore this construction provides a cryptanalysis-tolerant composition for commit hash functions.

The practical implication of our result is that candidate cryptographic hash functions may be cryptanalyzed against the requirements and properties of commit-hash function. Then, three candidate functions may be composed using the 2/3 composition, creating a simple, efficient and practical hybrid commit-hash function. We can then use this function for many applications requiring cryptographic hash functions, and in particular for commitment. The construction may not guarantee other properties associated with cryptographic hash functions, such as pseudo-random output.

2. The 2/3 Composition

In this section we provide a high-level overview of our results.

We begin with a high-level description of the 2/3 composition, providing intuitive arguments for its cryptanalysis-tolerance for the case of `standard` cryptographic hash functions (without any key or other input except the hashed string).

We then discuss briefly the commitment schemes and their relations to cryptographic hash functions, focusing on the variants related to this paper. We present a high-level description on the 2/3 composition for commitment schemes, and discuss the `standard` construction of commitment scheme from hash function. Finally, we introduce commit-hash functions and show how they can be used for typical applications of cryptographic hash functions, as well as for commitments; we also compare commit-hash functions to the related notions of oracle-hashing and Perfectly One Way (POW) hashing.

2.1. Simplified 2/3 Composition for Hash Functions

We first present a simplified version of the 2/3 composition, illustrated in Figure 2. Here, we construct a composite hash function *h* from three candidate `standard` cryptographic hash functions: h_0, h_1 and h_2. All of the hash functions have a single parameter (and in particular no key), as for MD5, SHA-1, RIPEMD-160 and most other standards for cryptographic hash functions; in fact one can think of each of the three functions as being one of these three standard hash functions.

While the 2/3 composition as illustrated in Figure 2 is not as trivial as the cascade and parallel compositions, it also has a pretty simple, logical structure. Recall that our goal is to provide cryptanalysis-tolerance for both confidentiality and collision-resistance (integrity) properties. Consider the composition as a directed graph; intuitively, we want the following properties, which can be easily formalized as properties of the graph:

- *Confidentiality:* each path from the message *m* to its hash *h(m)* must pass through (at least) two *different* hash functions, thereby applying the two functions *in cascade*. Since we assume two of the three functions ensure confidentiality, this ensures that each output is the result of a cascade involving at least one confidentiality-protecting hash function. Hence, from our trust in cascading of confidentiality-protecting functions, the 2/3 construction seems to protect confidentiality.
- *Collision-resistance (integrity):* removing the nodes corresponding to any of the three functions does not disconnect the graph. Hence, since we assume at least two of the three functions, say h_0, h_1 are collision-resistant, the output always contains a part which was computed by a path of collision-resistant functions. Hence, if we have a collision for *h*, i.e. two messages *m, m'* such that *h(m)=h(m')*, we can always `trace` this collision back along this path to find a collision for at least one of the collision-resistant functions.

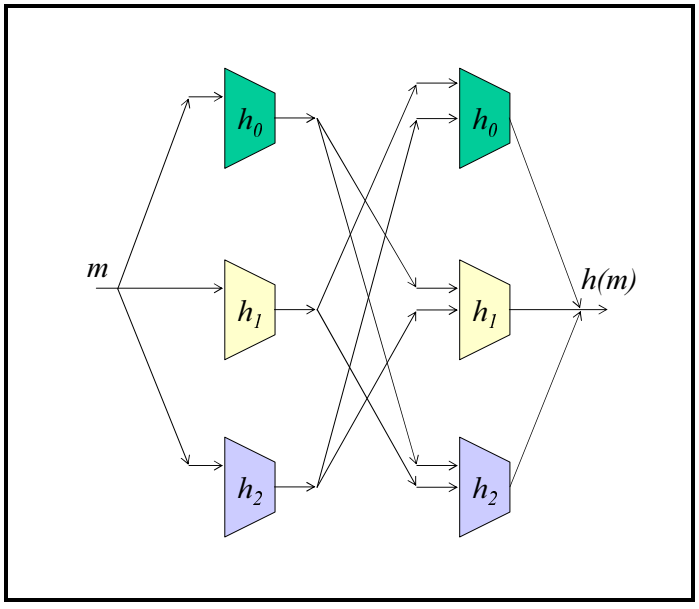

Figure 2: Simplified 2/3 Composition for `Standard` Hash Functions

We hope that above arguments will help the readers gain intuition for the 2/3 composition and its security, as well as explore similar constructions using the same

reasoning, e.g. generalizing the simple graph properties discussed. However, we wish to point out that these arguments are insufficient as proofs of security. Some of the reasons for that include:

- *Confidentiality:* it is very easy to see that even if all functions, independently, are one-way functions, than their 2/3 composition may not be one-way. For example suppose that h_1 copies the low-value half of its input bits and simply outputs them as the high-value half of its output bits, and h_2 does the opposite. Clearly, any input is completely revealed by the outputs of h_1 and h_2.
- *Collision-resistance:* We demonstrate the problem assuming that two of the functions are weakly collision resistant, i.e. it is hard to find a collision to a random input value (also called *2nd pre-image resistance*), as the standard stronger definition of collision-resistance is inapplicable to unkeyed hash functions. Suppose h_0 outputs the constant of all zeros. Both h_2 and h_1 are weakly collision-resistant, but have the property that for the small group of inputs whose lower-value half is all zero, the output is also uniformly zero (notice that such functions remain weakly collision resistant). Clearly the output of h is all zero for any input.

The reader may experiment with different variations on the collision-resistance and confidentiality properties; we did not come up with satisfying definitions for which the simplified construction of Figure 2, with deterministic hash functions, seems sufficient to ensure cryptanalysis-resiliency. This is not surprising as other important properties, such as (strong/any) collision resistance, are impossible for such `standard` hash functions, which has only one input (the message), without random input or key. Our results hold for commit-hash functions (defined later), which use random input and (non-secret) key.

2.2. Simplified 2/3 Composition for Commitment Schemes

Commitment schemes, like cryptographic hash functions, provide both confidentiality and integrity (binding, collision-resistance). We focus on simple, non-interactive commitment schemes, which we describe as a triplet of algorithms *{C,D,V},* standing for *Commit, Decommit* and *Validate* respectively. For provable security, we actually consider families of commitment schemes *{C[K],D[K],V[K]},* where a particular scheme is selected by a key *K;* but for simplicity of exposition, we usually omit *K* (except when it is relevant to the discussion).

To commit to a message *m,* the sender selects a random string *r* and computes commitment *c=C(m;r)* and decommitment *d=D(m;r).* Then the sender sends (only) *c* to the receiver. The *confidentiality* of the commitment scheme ensures that *c* does not expose any information about *m;* in particular the adversary cannot distinguish between the commitments of any two messages *m, m'* of its choice: *C(m;r)≈C(m';r').*

The sender sends the decommitment *d* to `open` the commitment; the receiver validates and recovers *m* by computing *V(c,d),* which returns *m* (or ⊥ for invalid *c,d*). The *integrity (binding, collision-resistance)* property of the commitment scheme prevents finding *c,d,d'* such that *V(c,d)=m , V(c,d')=m'* where *m* and *m'* are distinct and differ from ⊥. That is, the adversary cannot find a commitment value *c* which it can open in two different ways.

In Figure 3 we present a simplified 2/3 composition for commitment schemes. The intuitive arguments for its security are similar to these presented for hash functions.

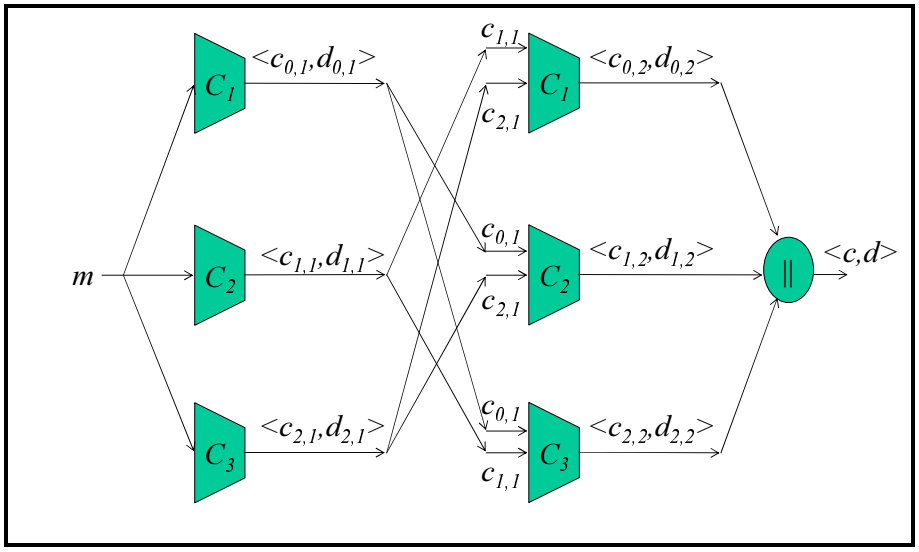

Figure 3: Simplified 2/3 Composition for Non-Interactive Commitment Schemes

2.3. Commit-Hash Functions and their 2/3 Composition

The previous subsections highlighted the similarity in the goals of cryptographic hash functions and of commitment schemes, as well as in the (simplified) 2/3 compositions for the two primitives. In fact, intuitively it seems that to commit we can just send *h(m||r),* namely hash of the message concatenated with a random string *r* (which we send with *m* to decommit); this is done in several applications. This construction is clearly not secure when using an arbitrary collision resistant (and one-way) hash function; e.g., the first few bits of *m* may be exposed, even when *h* is a one-way function. The construction is secure when using Perfectly One Way (POW) hash functions [CMR98], since the addition of the randomizer *r* ensures that the total input *m||r* has sufficient entropy. However, this is a wasteful solution, as it protects a random string (*r*) in the same way that it protects the message (*m*); notice that POW hash functions also use internal randomization (in addition to *r*).

For our purposes, it seems better to distinguish between the message *m* and the randomizer *r,* by considering them as distinct parameters to the hash function. For simplicity we assume that the both the randomizer and the (non-secret) key have length *k,* the `security parameter`.

Definition: A (Variable-Input Length) commit-hash function is a deterministic function $C[K](m;r): \{0,1\}^k \times \{0,1\}^* \times \{0,1\}^k \rightarrow \{0,1\}^l$. A Fixed-Input Length (FIL) commit-hash function is a deterministic function $C[K](m;r): \{0,1\}^k \times \{0,1\}^M \times \{0,1\}^k \rightarrow \{0,1\}^l$, with $M>l$.

When the value of the key *K* is fixed, we sometimes omit it and write simply *C(m;r).*

The random input *r* allows commit-hash functions to be used for commitment; specifically, commit by sending *C(m;r)* and open the commitment by sending *m, r.*

The *confidentiality* of commit-hash ensures that *c* does not expose any information about *m;* in particular the adversary cannot distinguish between the commitments of any two messages *m, m'* of its choice: $C(m;r) \approx C(m';r')$. Decommitment is by sending *(m,r),* and validation is by comparing the received commitment to the computed commitment $C(m;r)$, and outputting *m* if identical (and $\perp$ if not).

We say that a commit-hash $C(m;r)$ is *collision-resistant* if it is infeasible to find *m'* distinct from *m,* and any *r,r',* such that $C(m;r) = C(m';r')$. Trivially, if a commit-hash *C* is collision resistant, then the commitment scheme defined by it (as above) is also collision-resistant.

The 2/3 construction for Commit-Hash Functions is illustrated in Figure 4. Notice that for completeness, in this figure (and the definition) we explicitly included the (non-secret) key selecting the function. The construction is presented for compositions of VIL commit-hash functions. For composing FIL functions $C_i(m;r)$: $\{0,1\}^M \times \{0,1\}^k \rightarrow \{0,1\}^l$, with $M>3l$, add arbitrary $M-2l$ bits to the inputs of the second column.

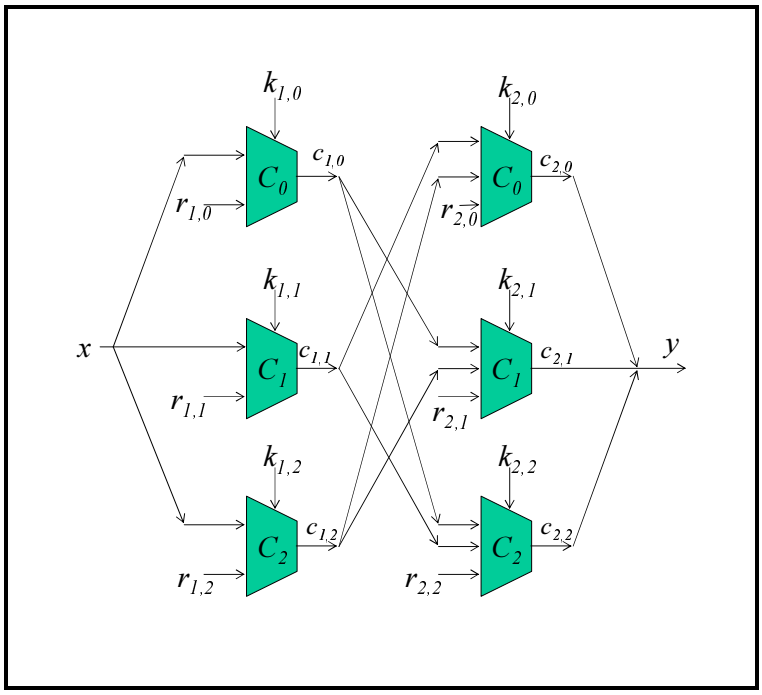

Figure 4: The 2/3 Commit-Hash Composition

Definition of the 2/3 commit-hash composition $\frac{2}{3}[C_i]$**.** For *i=0,1,2,* let *C* be VIL commit-hash functions. Let $r=\{r_{i,j}\}_{i=0,1,2;j=1,2} \in \{0,1\}^6$, $K=\{K_{i,j}\}_{i=0,1,2;j=1,2} \in \{0,1\}^6$. Let the *2/3 composition of {C},* denoted $\frac{2}{3}[C_i]$, be the VIL commit-hash function *C* defined by the following process:

Process C[K](m;r):

For i:=0,1,2 do $c_{i,1} := C[K_{i,1}](m;r_{i,1})$;

For i:=0,1,2 do $c_{i,2} := C[K_{i,2}](c_{(i+1 \bmod 3),1} || c_{(i-1 \bmod 3),1}; r_{i,2})$;

return $(c_{0,2} || c_{1,2} || c_{2,2})$;

The composite commit-hash function *C[K](m,r)* can be used as described above for commitment: commit by sending *C(m,r)* for random $r \in_R \{0,1\}^6$, open the commitment by sending *m,r,* and validate by comparing the received commitment *c* to the computed *C(m,r).* Notice this process relies on all parties knowing *K,* and on *C* being deterministic.

3. Security of Commit-Hash Functions and their 2/3 Composition

In this section we define the security properties for commit-hash functions, and show that the 2/3 composition provides cryptanalysis-tolerance for these properties, in the sense that the security properties of the composition are provably almost as well as the corresponding security properties of the 2nd-best of the composed functions (regardless of how weak the third function is).

We begin by defining, and then analyzing, the confidentiality properties. Then we define and prove the integrity (collision-resistance) properties.

3.1. Defining Confidentiality of Commit-Hash Functions

We begin by defining the confidentiality property for commit-hash functions and commitment schemes. Confidentiality relates only to the commit function *C* of the commitment scheme, hence for this discussion there is no difference between commit-hash functions and the corresponding commitment scheme. For simplicity we focus on *VIL* commit-hash functions; the extensions to FIL functions are obvious.

We require the commit-hash function to prevent an adversary from distinguishing between the outputs produced when the inputs are chosen from one of two distributions chosen by the adversary. This is a stronger requirement than the standard definitions of indistinguishability for encryption schemes, as defined by [BDJR97,Go02], which allows the adversary `only` to select two specific messages and distinguish the distributions resulting from encryptions of these messages. Our definitions are against a computationally bounded adversary; some definitions of commitment schemes, e.g. [DPP94,DPP98,HM96], ensure confidentiality even against computationally-unbounded adversary.

In order to allow analysis of compositions of functions, we found it helpful to consider adversaries with arbitrary auxiliary input, and bound the *extra* information that the adversary can learn from the commit-hash function, over the information the adversary can learn only from the auxiliary input. The use of auxiliary input is reminiscent of definitions of Zero-Knowledge with respect to auxiliary inputs, used to allow sequential composition of proofs [G01, section 4.3.4].

A *distinguishing adversary with auxiliary input* $Auxiliary_b$ *for VIL commit-hash function C* is a program with access to:

1. Oracle $Commit_{C,K,b}$ as defined below, to which the adversary provides a pair of poly-time sampling algorithms (S_0, S_1) and the length of the sample (in unary) *1* .
2. An oracle $Auxiliary_b$ with arbitrary input-output behavior, and access to bit *b*.

The oracle $Commit_{C,K,,b}$ returns a commitment, using key K, to a value sampled from sampling algorithm S_b. The adversary must also specify, in each query, the length n of the sample (in unary), which is provided to the sampling algorithm; the output of the sampling algorithms must always be of length M. The operation of the oracle $Commit_{C,K,,b}$ is defined as follows:

Oracle $Commit_{C,K,,b}(S_0,S_1,1^M)$
$s:=S_b(1^M)$; $s':=S_{b\oplus 1}(1^M)$;
If $|s'|\neq M$ or $|s|\neq M$ then return $\perp$; // *Adversary provided `cheating` S_0,S_1!*
r:=random element in $\{0,1\}^k$;
return $C_K(s;r)$;

Definition of guess with auxiliary input: Let C be a VIL commit-hash function, and let A be a distinguishing adversary for C with auxiliary input $Auxiliary_b$. The following process is called a *guess:*
Process $\mathbf{Guess}_{C,A,Auxiliary}(b)$
K:=random element in $\{0,1\}^k$;
return the output of $A(K)$ with access to oracles $Commit_{K,b}$ and $Auxiliary_b$;

The *exposure advantage* of *A for commit-hash function C using Auxiliary is:*
$\mathbf{Exp}_{C,A,Aux} = \Pr[\mathbf{Guess}_{C,A,Auxiliary}(1)=1] - \Pr[\mathbf{Guess}_{C,A,Auxiliary}(0)=1]$

Let A' be a program with access to *Auxiliary.* Define:
Process $\mathbf{Guess}_{A',Auxiliary}(b)$
return the output of A' with access to oracle $Auxiliary_b$;
The *exposure advantage of A' using Auxiliary is:*
$\mathbf{Adv}_{A',Aux} = \Pr[\mathbf{Guess}_{A',Auxiliary}(1)=1] - \Pr[\mathbf{Guess}_{A',Auxiliary}(0)=1]$

For any t,q,μ we define the *extra exposure* of C *over Auxiliary* as:
$\mathbf{Extra}_{C,Auxiliary}(t,q,\mu) = max_A \{ \mathbf{Adv}_{C,A,Auxiliary} \} - max_{A'} \{ \mathbf{Adv}_{A',Auxiliary} \}$
Where the maximum is over all adversaries A, A' having time complexity t, making at most q queries to the oracle, the sum of whose lengths is at most μ bits. In the rest of our discussion we omit the parameters t,q,μ; in our analysis and reductions, these parameters are changed only minimally or not at all.

Our assumption is that good commit-hash functions C would always add very small advantage (if at all), compared to any Auxiliary inputs (including none). Therefore we define the *maximal exposure due to commit-hash function C* as:

$$\mathbf{\Lambda}_C \equiv \max_{Aux} \mathbf{Extra}_{C,Aux}$$

3.2. Confidentiality of 2/3 Commit-Hash Composition

The 2/3 composition provides a commit-hash function whose maximal exposure is no more than four times that of the 2nd-best of the three component commit-hash functions. Without loss of generality, assume[1] that $\Lambda_{C_0} \leq \Lambda_{C_1} \leq \Lambda_{C_2}$. We prove:

[1] For simplicity, assume the same order $\Lambda_{C_0} \leq \Lambda_{C_1} \leq \Lambda_{C_2}$ holds for all relevant values of t,q,μ.

Theorem 1: Let $C=\frac{2}{3}[C_i]_{i=0,1,2}$ be the *2/3 composition* of commit-hash functions $\{C_0,C_1,C_2\}$, with corresponding maximal exposures $\Lambda_{C_0} \leq \Lambda_{C_1} \leq \Lambda_{C_2}$. Then $\Lambda_C \leq 4\Lambda_{C_1}$.

We begin our analysis by showing that the maximal advantage contributed by a parallel composition of commit-hash functions is at most the sum of the maximal advantage contributed by the two functions. Therefore we can bound the exposure due to parallel composition of commit-hash functions.

Lemma 1 (Parallel composition): Let $C^*[k,k'](x;r,r')=C[k](x;r)||C'[k'](x;r')$. Then $\Lambda_{C^*} \leq \Lambda_C + \Lambda_{C'}$.

Proof: Consider an arbitrary auxiliary *Aux,* and consider the maximal advantage it may provide together with *C:*

$$max_A \{ \mathbf{Adv}_{C,A,Aux} \} = max_A \{ \mathbf{Adv}_{A,Aux} \} + \mathbf{Extra}_{C,Aux}(t,q,\mu)$$
$$\leq max_A \{ \mathbf{Adv}_{A,Aux} \} + \Lambda_C$$

Consider now the combination of *Aux* and *C* as a new auxiliary *Aux',* namely

$$max_A \{ \mathbf{Adv}_{A,Aux'} \} = max_A \{ \mathbf{Adv}_{C,A,Aux} \} \leq max_A \{ \mathbf{Adv}_{A,Aux} \} + \Lambda_C.$$

Consider the maximal advantage of *Aux'* together with *C'* (i.e. of *Aux* with *C, C'*):

$$max_A \{ \mathbf{Adv}_{C',A,Aux'} \} = max_A \{ \mathbf{Adv}_{A,Aux'} \} + \mathbf{Extra}_{C',Aux'}(t,q,\mu)$$
$$\leq max_A \{ \mathbf{Adv}_{A,Aux} \} + \Lambda_C + \Lambda_{C'}$$

■

From this Lemma, we immediately deduce that we can consider, for the analysis of the exposure, a simplified composition where we replace the first two applications of C_1 and C_0 by a single commit-hash function corresponding to their parallel composition. Furthermore, notice that the Lemma does not depend on the fact that the input to the two composed functions is exactly identical; therefore we can also compose the applications of C_1 and C_0 in the second column. We can therefore analyze confidentiality (exposure) considering only the simplified composition of C_2 and C', where C' is the parallel composition of C_1 and C_0, as illustrated in Figure 5.

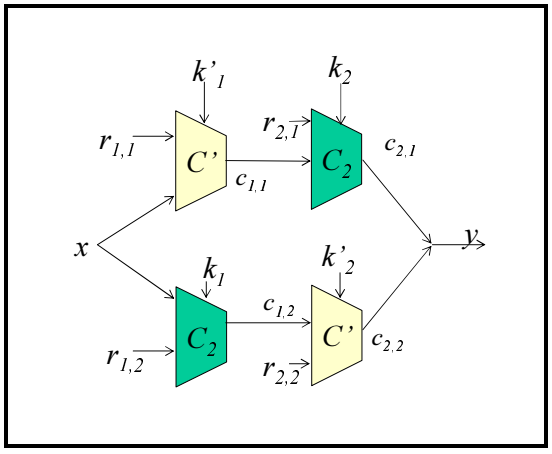

Figure 5: Simplified Commit-Hash Composition for Exposure Analysis

The next Lemma shows that, like other confidentiality primitives, a cascade of commit-hash functions exposes as little as the least exposing function in the cascade. This allows us to completely ignore the applications of C_2 (the commit-hash with the greater exposure) in our composition.

Lemma 2 (Cascade): Let $C[k_1,k_2](x;r_1,r_2)=C_2[k_2](C_1[k_1](x;r_1);r_2)$. Then

$\Lambda_C \leq min\{\Lambda_{C_1}, \Lambda_{C_2}\}$.

Proof (sketch): We first show that $\Lambda_C \leq \Lambda_{C_1}$. Let *A, Aux* be the adversary and auxiliary input achieving extra exposure advantage Λ_C for *C*. Define adversary A_1 for C_1 as follows: A_1 applies adversary *A,* implementing the *Commit* oracle for *A* as follows: whenever *A* calls *Commit,* then *A'* calls *Commit* for C_1, with the same inputs as *A* (in particular the same input distributions S_0,S_1); and when receiving the output *c*, A_1 returns to *A* the result of $C_2[k_2](c;r_2)$, where k_2, r_2 are selected randomly (by A_1). Clearly A_1 achieves the same advantage as *A*, and therefore $\Lambda_C \leq \Lambda_{C_1}$.

The proof that $\Lambda_C \leq \Lambda_{C_2}$ follows similarly, except that here A_2 specifies sampling algorithms S'_0,S'_1, by applying C_1 to the output of S_0,S_1 correspondingly, namely $S'_0(1^n)=C_1[k_1](S_0(1^n);r_1)$, where k_1, r_1 are selected randomly (by A_2). ■

By applying Lemma 2 to the simplified composition of Figure 5, we get a parallel composition of two invocations of *C'*, to which we apply Lemma 1, which completes the proof of Theorem 1. ■

3.3. Collision-resistance of 2/3 Commit-Hash Composition

We now define the integrity property of commit-hash functions, namely collision resistance. Again, for simplicity, we focus on VIL commit-hash functions. We slightly modify the definitions of *any collision resistance* from [BR97] and *collision resistance* of [Da89], to accommodate the additional input of a randomizer to commit-hash functions. Namely, a *collision* for commit-hash function *C* with key *K* is a set of two pairs, $\{<m,r>, <m',r'>\}$ such that $m \neq m'$ and $C[K](m; r)=C[K](m'; r')$. Notice that the values of the randomizers *r, r'* are irrelevant to a collision. The crux of the collision-resistance property of the 2/3 composition follows from the following simple Lemma. For simplicity we count computations of the component commit-hash functions as one computer operation.

Lemma 3: Let $C=\frac{2}{3}[C_i]_{i=0,1,2}$ be the *2/3 composition* of commit-hash functions $\{C_0,C_1,C_2\}$. Suppose $m \neq m'$ but $C[K](m;r)=C[K](m';r')$ for some r,r'. There is an algorithm that, given m,m',r,r' and *K,* outputs (in 24 operations) a collision for C_i with key $k_{1,i}$ or $k_{2,i}$, for at least *two* of $i \in \{0,1,2\}$.

Proof: The algorithm simply computes all the internal values of the composition for both (m,r) and (m',r') and looks for a collision. Clearly this takes less than 24 computer operations (considering an evaluation of each function as one operation). It remains to show that this provides the required collision.

We use the notations in Figure 4, adding an apostrophe when referring to computation of $C[K](m';r')$. Since $m=m'$ it follows that $c_{2,0}= c'_{2,0}$, i.e. $C_0[k_{2,0}](c_{1,1}||c_{1,2};r_{2,0}) =C_0[k_{2,0}](c'_{1,1}||c'_{1,2};r'_{2,0})$.

If $c_{1,1}||c_{1,2}=c'_{1,1}||c'_{1,2}$, there is a collision for *both* C_1 and C_2. Assume otherwise, i.e. $c_{1,1}||c_{1,2}\neq c'_{1,1}||c'_{1,2}$; this gives a collision for C_0.

Repeat this analysis, now beginning with the fact that $c_{2,1}= c'_{2,1}$, i.e. $C_1[k_{2,1}](c_{1,1}||c_{1,0};r_{2,1}) =C_1[k_{2,1}](c'_{1,1}||c'_{1,0};r'_{2,1})$. From the same arguments, we show that either there is a collision for *both* C_0 and C_2, *or* there is a collision for C_1. In total we showed collisions for at least two of the functions. ■

An adversary A is said to *find collisions in* commit-hash function C *with probability* ε, *using time* t, if its running time is at most t and the probability that, given $K\in_R\{0,1\}^k$, A outputs a collision for C with key K, is at most ε. The probability is taken over K and over A's coin flips.

A commit-hash function C is said to be $\varepsilon(t)$*-collision-resistant* if for every ε, t, no adversary A can find collisions in C with probability ε using time t.

We can now prove that the 2/3 composition provides cryptanalysis-tolerance for the collision-resistance property. For simplicity we assume that generation of a random number takes a single computer operation.

Theorem 2: Let $C=\frac{2}{3}[C_i]_{i=0,1,2}$ be the *2/3 composition* of commit-hash functions $\{C_0,C_1,C_2\}$. If two of $\{C_0,C_1,C_2\}$ are $\varepsilon(t)$*-collision-resistant,* then C is $\frac{3}{2}\varepsilon(t+24)$ *collision-resistant.*

Proof: Standard reduction argument. Given an adversary A which find collisions in commit-hash function C with probability $\varepsilon'(t)$, we invoke it, selecting random keys and randomizers for all the functions (this requires at most 24 operations). Whenever A finds a collision, we use Lemma 3, to find collisions for at least two of $\{C_0,C_1,C_2\}$ (with at most 24 additional operations). Therefore, we found a collision for one of the two $\varepsilon(t)$*-collision-resistant* commitment-hash functions. ■

Together with Theorem 1, this proves:

Theorem 3: The 2/3 commit-hash composition ensures cryptanalysis-tolerance.

4. Conclusions and Questions

We presented *commit-hash functions*, a hash function which is trivial to use for commitment schemes. We also showed the 2/3 composition provides cryptanalysis-tolerance for commit-hash functions.

There are many directions for further research based on the current work; some may be trivial extensions and others may be more challenging. In particular:

1. Implementation of Commit-hash functions from simpler primitives, e.g. can we construct them from one-way functions (without losing too much security)? One natural candidate is to implement commit-hash from Perfectly One Way (POW) hash functions, which can be

constructed from weak cryptographic primitives, e.g. from any one-way permutation [CMR98].

2. Can we use commit-hash functions to construct secret-key cryptographic functions, in particular MAC, following previous works on constructing MAC functions from hash functions, in particular the HMAC and NMAC constructions of [BCK96]?
3. Does the 2/3 composition, or variation of it, hold for other variants of commitment schemes, in particular non-malleable commitment schemes [DDN91] and UC commitment schemes [CF01]?
4. Can we provide better cryptanalysis-tolerant compositions for hash functions and commitment schemes? Of particular interest is whether it is possible to create a `1-out-of-2` construction. Of course, a `6-out-of-9` composition is trivial…

References

[ABCV98] B. Aiello, M. Bellare, G. Di Crescenzo, and R. Venkatesan, Security amplification by composition: the case of doubly-iterated, ideal ciphers, Proc. of CRYPTO 98.

[ADR02] Jee Hea An, Yevgeniy Dodis and Tal Rabin, On the Security of Joint Signature and Encryption, in Theory and Application of Cryptographic Techniques, pp. 83-107, 2002.

[BCK96] Bellare, M., Canetti, R., and Krawczyk, H., "Keyed Hash Functions for Message Authentication", Advances in Cryptology -- CRYPTO~96 Proceedings, Lecture Notes in Computer Science, Springer-Verlag Vol. 1109, N. Koblitz, ed, 1996, pp. 1-15.

[BCK96F] M. Bellare, R. Canetti, and H. Krawczyk, Pseudorandom functions revisited: The cascade construction and its concrete security. Proceedings 37th Annual Symposium on the Foundations of Computer Science, IEEE, 1996.

[BDJR97] M.Bellare, A.Desai, E.Jokipii, P.Rogaway: A Concrete Security Treatment of Symmetric Encryption, Proceedings of the 38th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 394--403, 1997. Revised version at http://www-cse.ucsd.edu/users/mihir/papers/sym-enc.html.

[BKR94] M. Bellare, J. Kilian and P. Rogaway, "The security of cipher block chaining", Journal of Computer and System Sciences, Vol. 61, No. 3, Dec 2000, pp. 362--399. Extended abstract in Advances in Cryptology - Crypto 94 Proceedings, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed, Springer-Verlag, 1994.

[BR97] Mihir Bellare and Phillip Rogaway, Collision-Resistant Hashing: Towards Making UOWHFs Practical, Extended abstract was in Advances in Cryptology- Crypto 97 Proceedings, Lecture Notes in Computer Science Vol. 1294, B. Kaliski ed, Springer-Verlag, 1997. Full paper available at http://www.cs.ucsd.edu/users/mihir/papers/tcr-hash.html.

[C97] Ran Canetti, "Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information", Advances in Cryptology – Crypto 97 Proceedings, pp. 455-469, 1997.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In J. Kilian, editor, Advances in Cryptology - Crypto 2001, pages 19-40, Berlin, 2001. Springer-Verlag. Lecture Notes in Computer Science Volume 2139.

[CGHN97] Ran Canetti, Rosario Gennaro, Amir Herzberg and Dalit Naor, Proactive Security: Long-term Protection Against Break-ins, RSA CryptoBytes, Volume 3, No. 1, 1997.

[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.

[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.

[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] Herzberg, A., Luby, M., "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.

[NY89] Moni Naor, Moti Yung: Universal One-way Hash Functions and their Cryptographic Applications; 21st Symposium on Theory of Computing (STOC), 1989, ACM, New York, pp. 33-43.

[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.

[R92] Ronald Rivest, "The MD5 message digest algorithm," Internet RFC 1321, 1992, http://theory.lcs.mit.edu/rivest/Rivest-MD5.txt.

[Sh79] Adi Shamir, How to Share a Secret, Communications of the ACM 22, 1979, pp. 612--613.