Multicollision Attacks on Some Generalized Sequential Hash Functions

M. Nandi and D. R. Stinson
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
{m2nandi, dstinson}@cs.uwaterloo.ca

May 12, 2006

Abstract

A multicollision for a function is a set of inputs whose outputs are all identical. A. Joux showed multicollision attacks on the classical iterated hash function. He also showed how these multicollision attacks can be used to get a collision attack on a concatenated hash function. In this paper, we study multicollision attacks in a more general class of hash functions which we term "generalized sequential hash functions". We show that multicollision attacks exist for this class of hash functions provided that every message block is used at most twice in the computation of the message digest.

1 Introduction

In this paper, we discuss multicollision attacks on generalized sequential hash functions (a precise definition of this class of hash functions will be given in Section 3). A multicollision is a generalized notion of collision on a function. A *collision* on a function $g : X \to Y$ is a doubleton subset $\{x, y\} \subseteq X$ such that $g(x) = g(y)$. For an integer $r \geq 2$, an *r-way collision* (or *r-multicollision*) on a function, $g(\cdot)$, is an r-subset $\{x_1, \ldots, x_r\} \subseteq X$ such that $g(x_1) = g(x_2) = \ldots = g(x_r) = z$ (say). The common output value, z, is known as the *collision value* for this r-way collision set. An *r-way collision* or *r-multicollision attack* is an algorithm which finds an r-multicollision set with some non-negligible probability.

The *birthday attack* for r-way collisions has time complexity $\Theta(|Y|^{(r-1)/r})$. When $r = 2$, the time complexity for finding a collision is $\Theta(|Y|^{1/2})$. The time complexity of an attack algorithm is usually proportional to the number of computations of the underlying function, g, required to get a multicollision set. From now on, we will use the word "complexity" to mean time complexity, as measured by the number of computations of the underlying function.

A *classical iterated hash function* [4] [18], $H : \{0,1\}^* \to \{0,1\}^n$, is based on a *compression function*, $f : \{0,1\}^n \times \{0,1\}^{n'} \to \{0,1\}^n$. Here, we compute the value of H in the following way;

1. Given a message, we first apply some good *padding rule*, which includes some representation of the length of the message, so that the length of the padded input becomes a multiple of n' (see [4], [17] and [18] for more details). To analyze the hash function in a simpler way,

we will usually ignore the padding rule. For example, a padding rule is irrelevant if we are concerned with messages of some fixed length.

2. Let $M = m_1 \parallel \cdots \parallel m_\ell$ be a *padded message*[1] with $|m_i| = n'$ for $1 \leq i \leq \ell$, and let h_0 be some fixed n-bit *initial value*. Then the classical iterated hash function, H, based on the compression function f and the initial value h_0, is defined as follows: $H(M) = h_l$, where $h_i = f(h_{i-1}, m_i)$, $1 \leq i \leq \ell$. For each $1 \leq i < \ell$, h_i is known as an *intermediate hash value*, and $H(M)$ is the *output hash value* (also known as a *message digest*).

The above-described method is the most frequently used technique for the design of practical hash functions.

Recently, A. Joux [12] found an algorithm to construct a 2^r-multicollision set on a classical iterated hash function, having time complexity $\mathrm{O}(r\,2^{n/2})$, which is a considerable improvement over the birthday attack (see Theorem 2.2). This multicollision attack can be considered as a weakness of the iterated hash function design because of the following reasons:

1. Joux also showed how the multicollision attack can be used to construct a collision attack, which is better than the birthday attack, on the concatenated hash function $H \parallel G$ where H is the classical hash function and G is any hash function.

2. There are some other practical applications where multicollision secure hash functions are required. These include the micropayment scheme Micromint [22], the identification scheme of Girault and Stern [8], the signature scheme of Brickell *et al* [2], etc.

3. Multicollision secure hash function design is an interesting fundamental question, because a function having an efficient multicollision attack gives evidence of the non-randomness of the function.

1.1 Our Contribution and Related Work

In light of the above discussion, it is an interesting question to find a good design technique for hash functions that are secure against multicollision attacks. In this paper, we provide some negative answers to this question. We study a certain generalization of the classical iterated hash function in which every message block is used up to two times in the computation of a message digest. This generalization is a natural way to attempt to prevent Joux's attack from succeeding, and it is therefore worthwhile to study this approach in detail. Unfortunately, we show that these generalized sequential hash functions also have efficient multicollision attacks. We find 2^r-multicollision attacks with expected complexity $\mathrm{O}(r^2 \ln r(n + \ln\ln r)2^{n/2})$. Thus, we rule out a natural and large class of hash functions as candidates for multicollision secure hash functions.

The attacks presented in this paper were first described in a more informal way in the PhD thesis of the first author [19] and in the unpublished manuscript [20]. In the current paper, we are presenting complete descriptions of the attacks as well as precise complexity analyses.

In [19] and [20], similar attacks on so-called "tree-based hash functions" were also pointed out. The attacks assumed that every message block is used at most twice (similar to the assumption we make regarding generalized sequential hash functions). Recently, our attacks were extended and generalized by Hoch and Shamir [11]. Their paper considers generalized sequential or tree-based

[1] Sometimes we may write M as an ℓ-tuple, i.e., $M = (m_1, \ldots, m_\ell)$.

hash functions that have a fixed *expansion rate*, which indicates the maximum number of times a message block is processed in the evaluation of a message digest (our attacks concern hash functions with expansion rate 2).

1.2 Organization of Our Paper

In Section 2 introduction birthday attacks, Joux's attack and its applications. In Section 3, we provide a definition of a general class of iterated hash functions. The "generalized sequential hash functions" we study in this paper are a special case of this definition. Section 4 defines some useful terminology and results on sequences and partial orders. Then, in Section 5, we describe our attacks and provide a detailed complexity analysis. Finally, Section 6 is a brief conclusion.

2 Preliminaries

In this section, we give a brief introduction to the birthday attack, which is the basis for the attacks to be used throughout this paper. We also state some recent results on multicollision attacks which motivate the rest of the paper. Namely, we discuss Joux's multicollision attack on classical iterated hash functions. Finally we give some simple but important applications of multicollision attacks.

2.1 The Birthday Attack

A hash function usually has two main components: a compression function, $f : \{0,1\}^n \times \{0,1\}^{n'} \to \{0,1\}^n$, and a method to extend the domain of the compression function into $\{0,1\}^*$. The second component is also known as the "design of iteration" as we generally iterate the compression function several times. Throughout this paper, we consider only attacks which treat the underlying compression function, f, as a black-box. Thus, the attacker is not exploring any internal structure of f. The attacker can only make some queries to the function f, and, based on the responses of the queries, he finally outputs some value or values. Here a *query* denotes an input to f, say x, and the *response* denotes the output, $y = f(x)$. This type of attack can be applied to any compression function, and hence it mainly points out the security of the design of iteration.

We recall that the complexity of a r-multicollision attack algorithm is the number of queries of f required to get a r-multicollision. A natural (and the most popular) attack is the *birthday attack*. It is well-known that the standard birthday attack (which finds a 2-way collision) has complexity $\Theta(2^{n/2})$ when message digests are n bits in length. In a birthday attack on a function $g : X \to Y$, we assume that the function g is a *random oracle*; i.e., every output value $g(x)$ is chosen uniformly at random from Y. Random oracles are the usual model for hash functions which can be accessed a black-box manner.

For the standard birthday attack, we will make use of the following bound (see [1, 17, 25, 26] for a more detailed discussion).

Theorem 2.1. (Complexity of the Standard Birthday Attack) *For a random oracle $g : X \to \{0,1\}^n$, the birthday attack with complexity q finds a 2-way collision with probability roughly equal to $1 - e^{-q^2/2^{n+1}}$.*

The following algorithm describes the generalized birthday attack of complexity q for (possibly) finding an r-multicollision of the function g, based on making q queries to the function g.

Generalized Birthday Attack	
Input:	A random oracle $g : X \to \{0,1\}^n$; complexity q; and desired multicollision size r.
1.	Choose $x_1, \ldots, x_q$ uniformly at random from the domain X and compute $y_i = g(x_i)$ for $1 \leq i \leq q$.
2.	If there is a subset $\mathcal{C} \subseteq \{x_1, \ldots, x_q\}$ of size r such that $\mathcal{C}$ is an r-way multicollision subset for the function g, then return $\mathcal{C}$. Otherwise return the output "failure".

We note that the standard birthday attack is just the special case $r = 2$ of the generalized birthday attack. The next theorem gives an estimate of the complexity of the generalized birthday attack in finding an r-multicollision with some specified probability p.

Theorem 2.2. (Complexity of the Generalized Birthday Attack)
For a random oracle $g : X \to \{0,1\}^n$, the birthday attack with complexity q finds an r-way collision with probability p provided that $q \approx (r!)^{1/r}\, 2^{n(r-1)/r}\, (\ln(1/(1-p)))^{1/r}$. For fixed p, the complexity is $\Theta(r\, 2^{n(r-1)/r})$. For fixed p and r, the complexity is $\Theta(2^{n(r-1)/r})$.

Proof. We use an estimate given by Diaconis and Mosteller [6]. They state that the generalized birthday attack having complexity q finds an r-way collision with probability p, where

$$q\, e^{-q/(r2^n)} \left(1 - \frac{q}{(r+1)2^n}\right)^{-1/r} \approx \left(2^{n(r-1)} r! \ln\left(\frac{1}{1-p}\right)\right)^{1/r}. \qquad (1)$$

For values of n and r of cryptographic interest, the left side of (1) is essentially equal to q. Using the facts that $(r!)^{1/r}$ is $\theta(r)$ and $\lim_{r\to\infty}(\ln(1/(1-p)))^{1/r} = 1$, the stated results follow. □

2.2 Joux's Multicollision Attack

In a recent paper by Joux [12], it was shown that there is a 2^r-way collision attack for the classical iterated hash function based on a compression function, $f : \{0,1\}^{n+n'} \to \{0,1\}^n$, where the attack has complexity $\mathrm{O}(r\, 2^{n/2})$. This complexity is much less than the complexity for the generalized birthday attack (see Theorem 2.2).

Here is the basic idea of Joux's attack. Consider the set of n-tuples $\{0,1\}^n$. We use the notation $h \xrightarrow{m} h'$ (a labeled arc) to mean $f(h, m) = h'$, where $|h| = |h'| = n$ and $|m| = n'$. The strategy of Joux's attack is to first find r successive collisions by performing r successive birthday attacks, as follows:

$$\begin{array}{lcll}
h_0 \xrightarrow{m_1^1} h_1 & \text{and} & h_0 \xrightarrow{m_1^2} h_1 & \text{for some } h_1\text{, where } m_1^1 \neq m_1^2 \\
h_1 \xrightarrow{m_2^1} h_2 & \text{and} & h_1 \xrightarrow{m_2^2} h_2 & \text{for some } h_2\text{, where } m_2^1 \neq m_2^2 \\
\quad\vdots & & & \\
h_{r-1} \xrightarrow{m_r^1} h_r & \text{and} & h_{r-1} \xrightarrow{m_r^2} h_r & \text{for some } h_r\text{, where } m_r^1 \neq m_r^2.
\end{array}$$

In other words, for $1 \leq i \leq r$, we apply a birthday attack to find $m_i^1 \neq m_i^2$ such that

$$f(h_{i-1}, m_i^1) = f(h_{i-1}, m_i^2) = h_i,$$

for some h_i. Then the set

$$\{m_1^1, m_1^2\} \times \{m_2^1, m_2^2\} \times \cdots \times \{m_r^1, m_r^2\}$$

is a 2^r-way collision set.

Of course, the birthday attack is itself a probabilistic attack. From Theorem 2.1, we see that each birthday attack succeeds with probability approximately equal to 0.4 and therefore the expected number of applications of each birthday attack is about 2.5. We do not carry out the ith birthday attack until the $(i-1)$st attack has succeeded. Therefore the total expected number of birthday attacks required in Joux's attack is about $2.5r$, which is just $\mathrm{O}(r)$. The complexity of each birthday attack is $2^{n/2}$, so the overall expected complexity of Joux's attack is $2.5r\,2^{n/2}$, which is $\mathrm{O}(r\,2^{n/2})$.

2.3 Applications of Multicollision Attacks

The birthday attack is feasible for small sized output hash values. To make the birthday attack infeasible, one simply specifies a large output hash value. There are many approaches of designing hash functions having large output hash values based on secure block ciphers; see [9], [10], [15], [14], [21], [24]. However, most of these designs were proven to be insecure; see [9], [15] [14], [21], [24]. Recently, Hirose [10] designed secure double length hash functions based on secure block ciphers. But the efficiency of the design is fairly low.

A natural and efficient approach to produce large output hash values is the concatenation of several smaller output hash values. For example, given two classical iterated hash functions, H and G, one can define a hash function $H(M) \parallel G(M)$. This idea has been frequently used because it is efficient and simple to implement. However, due to the attacks of Joux [12], there exists a collision attack that is more efficient than than the birthday attack. The complexity of the attack is roughly the maximum of the complexity of the multicollision birthday attack on H and the complexity of the standard birthday attack on G.

We briefly describe the attack (see [12] for more details). Let H and G have output hash values of n_H and n_G bits in length, respectively.

1. By using Joux's multicollision attack, find $2^{n_G/2}$ messages which have common output hash value (say h^*) on H.

2. Find two messages, say M and N where $M \neq N$, which are members of the set of $2^{n_G/2}$ messages found in step 1, such that they have same output hash value (say g^*) on G. Note that we expect to be able to find a collision on an n_G-bit function from a set of $2^{n_G/2}$ messages using the standard birthday attack.

Thus, we have $H(M) \parallel G(M) = H(N) \parallel G(N) = h^* \parallel g^*$. The overall complexity of this attack is $\mathrm{O}(n_G\,2^{n_H/2} + 2^{n_G/2})$. Note that we only assume that H is a classical iterated hash function; G can be any hash function at all.

Remark 2.1. As mentioned previously, we ignore the padding that includes the binary representation of the length of the inputs. Note that, even if we included the padding, it does not affect the above attack, as the multicollision sets consist of messages of equal length.

3 A General Class of Hash Functions

We have seen in Section 2.2 that the classical iterated hash function is vulnerable to a multicollision attack. Thus one cannot use the classical iterated hash function if multicollision secure hash functions are needed. There are some other disadvantages of using classical iterated hash functions. For example, Kelsey and Schneier [13] have found a generic second preimage attack that is better than the birthday attack.

To fix all these problems, one can try to use some suitable variant of the classical iterated hash function. We note that, recently, Lucks [16] designed a hash function that is secure against multicollision attack. In his construction a "wide" compression function is used. The hash function is proven to be secure if the compression function and the output transformation are both random oracles.

Alternatively, one might consider a modification of the classical iterated hash function where message blocks are used more than once. Another approach is to use a parallel design, which is characterized by a directed tree; see [23]. One can also combine these two approaches.

These types of generalized hash functions could be considered as an alternative to the classical iterated hash function since Joux's attack cannot be applied. For example, the hash function $H'(M) = H(H(IV, M), M)$ uses each message block twice. Here H denotes the classical iterated hash function. We call this hash function a *doubly iterated hash function* as it uses the classical iteration technique twice. Obviously, Joux's attack can not be applied directly to this hash function. Thus it is worthwhile to study this class in more detail.

Remark 3.1. The idea of "cycling through the message blocks twice" is apparently due to Davies and Price [5]. Their construction uses an encryption function instead of a compression function, and it is susceptible to meet-in-the-middle attacks (see [3, 7]). The meet-in-the-middle attacks exploit the invertibility of an encryption function and cannot be applied in the context we are considering.

We define a very general class of hash functions. Let $f : \{0,1\}^N \rightarrow \{0,1\}^n$ be a compression function. A hash function H from the class behaves in the following manner:

1. It invokes f a finite number of times.
2. The entire output of any intermediate invocation (not the final invocation) is fed into the input of other invocations of f.
3. Each bit of the message, M, is fed into at least one invocation of f.
4. The output of the final invocation is the output of the hash function, H.

We define a general class $\mathcal{D}$ of hash functions satisfying the above conditions. We will assume that our message has the form $M = m_1 \parallel \cdots \parallel m_\ell$, where each m_i is a message block that is an n'-bit string. We also assume that we have a fixed set of *initial values*, denoted $v_1, v_2, \cdots$, each of which is an n-bit string. Every input to f is of the form $h \parallel x$. Each h is the concatenation of d' n-bit strings, each of which is a previously computed output of f or an initial value; and each x is the concatenation of d message blocks. We will require that $d' \geq 0$, $d \geq 0$ and $n'd + nd' = N$.

Then we can specify the computation of the hash function by a list of triples

$$L = \{(h_i, x_i, y_i) : 1 \leq i \leq s\},$$

where the following conditions hold for all i:

1. $f(h_i \parallel x_i) = y_i$,
2. $h_i = h_i^1 \parallel \cdots \parallel h_i^{d'}$,
3. $h_i^j \in \{v_1, v_2, \dots\} \cup \{y_1, \dots, y_{i-1}\}$,
4. $x_i = x_i^1 \parallel \cdots \parallel x_i^d$, and
5. $x_i^j \in \{m_1, \dots, m_\ell\}$.

Each y_i is an *intermediate hash value* and y_s is the *output hash value.* Note that the values of d and d' do not have to be constant; they may depend on i. However, $nd + n'd' = N$ must always hold.

3.1 Generalized Sequential Hash Functions

In this paper, we consider a special (but still quite general) subclass of $\mathcal{D}$, which are termed *generalized sequential hash functions* and denoted by $\mathcal{S}$. In the class $\mathcal{S}$ of generalized sequential hash functions, we have $d = d' = 1$ for all i and $N = n' + n$. Define a sequence $\alpha = \langle \alpha_1, \cdots, \alpha_s \rangle$ where $\alpha_i \in \{1, 2, \dots, \ell\}$. Let $h_1 = v_1$ (an initial value), $h_i = y_{i-1}$ for all $i \geq 2$, and let $x_i = m_{\alpha_i}$ for all $i \geq 1$. Hence, we can express the computation in the form

$$h_i = f(h_{i-1}, m_{\alpha_{i-1}}),\ 2 \leq i \leq s+1,$$

where h_1 is an initial value and h_{s+1} is the final hash value.

We can present this hash function diagrammatically, as follows:

$$h_1 \xrightarrow{m_{\alpha_1}} h_2 \xrightarrow{m_{\alpha_2}} h_3 \xrightarrow{m_{\alpha_3}} \quad \cdots \quad \xrightarrow{m_{\alpha_{s-1}}} h_s \xrightarrow{m_{\alpha_s}} h_{s+1}.$$

Note that each message block must be used at least once in the above computation. In the case of the classical iterated hash function, however, we have $\alpha_i = i$ for all i, and $s = \ell$. Also, in the classical iterated hash function, each message block is used exactly once.

To define a hash function with arbitrary domain $(\{0,1\}^N)^*$, say $H : (\{0,1\}^N)^* \to \{0,1\}^n$, we have a sequence of sequences $\langle \alpha^1, \alpha^2, \cdots \rangle$ such that $H(M)$ is defined based on the sequence α^ℓ, where α^ℓ contains elements from $\{1, \dots, \ell\}$, whenever M is an ℓ-block message. Also, each sequence α^ℓ should use every element in $\{1, \dots, \ell\}$.

4 Some Terminologies on Sequences and Partial Orders

Consider a fixed, finite sequence $\alpha = \langle \alpha_1, \alpha_2, \cdots, \alpha_s \rangle$ of *symbols*, where each symbol is an element of the *symbol set* $\mathcal{S} = \{1, \dots, \ell\}$ and every symbol occurs at least once in the sequence. The *length* of the sequence α is s and it is denoted by $|\alpha|$. The *index set* of the sequence α is $\{1, \dots, s\}$.

For a sequence of distinct indices, i.e., $I = \langle i_1, \dots, i_k \rangle$, $\alpha(I)$ denotes the sequence $\langle \alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_k} \rangle$. We also write this sequence as $\alpha\langle i_1, \dots, i_k \rangle$. The *interval* $[i, j]$ is defined to be the sequence $\langle i, i+1, \dots, j \rangle$. An *initial interval* is an interval of the form $[1, j]$. For $1 \leq i < j \leq |\alpha|$, we define $\alpha[i, j] = \langle \alpha_i, \alpha_{i+1}, \dots, \alpha_j \rangle$.

We next define a relation on the symbol set $\mathcal{S}$. For $x, y \in \mathcal{S}$, $x \neq y$, define $x \prec y$ if every occurrence of x in α precedes every occurrence of y in α. It is easy to see that the relation "$\prec$" is antisymmetric and transitive; hence "$\prec$" is a *partial order*.

Two symbols $x \neq y$ are *incomparable* with respect to a partial order "$\prec$" on a finite set X if it is not the case that $x \prec y$ or $y \prec x$. A list of symbols $x_1, \ldots, x_d$ is a *chain* if $x_1 \prec x_2 \prec \cdots \prec x_d$. A set of chains is a *chain decomposition* if the chains are disjoint and their union is X.

Later, we will use the classical result known as Dilworth's Theorem, which applies to any partial order.

Theorem 4.1. (Dilworth's Theorem) *Suppose that "$\prec$" is a partial order on a finite set X. Then the maximum number of mutually incomparable elements in X is equal to the minimum number of chains in any chain decomposition of* $(X, \prec)$.

We denote the maximum number of elements in a chain by $\mathrm{maxchain}(X)$. For our partial order defined with respect to a sequence α, this quantity is denoted by $\mathrm{maxchain}(\alpha)$. Obviously, if there are k elements which appear exactly once in the sequence α, then $\mathrm{maxchain}(\alpha) \geq k$.

We now consider some examples illustrating the above definition and terminologies. Later, we also show multicollision attacks on the generalized sequential hash functions based on these sequences.

Example 4.1. Let $\Psi^{(1,\ell)} = \langle 1, 2, \ldots, \ell \rangle$ (the sequence for the classical iterated hash function). It is easy to see that $1 \prec 2 \prec \cdots \prec \ell$ and hence $\mathrm{maxchain}(\Psi^{(1,\ell)}) = \ell$. ■

Example 4.2. Let $\Psi^{(2,\ell)} = \langle 1, 2, \ldots, \ell, 1, 2, \ldots, \ell \rangle$. The *doubly iterated hash function* is based on the sequence $\Psi^{(2,\ell)}$. It is easy to observe that there is no chain of length two in the sequence $\Psi^{(2,\ell)}$ and hence $\mathrm{maxchain}(\Psi^{(2,\ell)}) = 1$. ■

Example 4.3. Let $\Theta^\ell = \langle 1, 2, 1, 3, 2, 4, 3, \cdots, \ell-1, \ell-2, \ell, \ell-1, \ell \rangle$. Thus, for example, we have

$$\begin{aligned} \Theta^3 &= \langle 1, 2, 1, 3, 2, 3 \rangle, \\ \Theta^4 &= \langle 1, 2, 1, 3, 2, 4, 3, 4 \rangle, \quad \text{and} \\ \Theta^5 &= \langle 1, 2, 1, 3, 2, 4, 3, 5, 4, 5 \rangle. \end{aligned}$$

Here, it is easy to see that

$$\mathrm{maxchain}(\Theta^\ell) \geq \left\lfloor \frac{\ell+1}{2} \right\rfloor,$$

because

$$1 \prec 3 \prec \cdots \prec \ell$$

(if ℓ is odd) or

$$1 \prec 3 \prec \cdots \prec \ell - 1$$

(if ℓ is even) are chains. To prove that these are chains, consider the partition of the index set into intervals

$$[1, 3], [4, 7], [8, 11], \ldots [2\ell - 2, 2\ell].$$

Assume that ℓ is odd (the result for even ℓ can be proved similarly). Now one can see that $1 \in \Theta^\ell[1, 3]$, $3 \in \Theta^\ell[4, 7]$, … , $\ell \in \Theta^\ell[2\ell - 2, 2\ell]$. These elements do not appear in other intervals. Thus $1 \prec 3 \prec \cdots \prec \ell$ is a chain for the sequence Θ^ℓ.

In fact, we have $\text{maxchain}(\Theta^\ell) = \lfloor \frac{\ell+1}{2} \rfloor$. This can be proven as follows: For any $\lfloor \frac{\ell+1}{2} \rfloor + 1$ elements from $\{1, \dots, \ell\}$ there are two consecutive symbols (by applying the pigeonhole principle), say i and $i+1$, and hence there is a subsequence $\langle i, i+1, i \rangle$ of Θ^ℓ. Thus no $\lfloor \frac{\ell+1}{2} \rfloor + 1$ elements can form a chain. ■

For a sequence α on symbols $\{1, \dots, \ell\}$ and a symbol $x \in \{1, \dots, \ell\}$ we define

$$\text{freq}(x, \alpha) = |\{i : \alpha(i) = x\}|.$$

Sometimes we write this as $\text{freq}(x)$ and we call it the *frequency of* x. This value denotes the number of times x appears in the sequence α. We also write $\text{freq}(\alpha)$ (*frequency of the sequence*) for the maximum frequency of any element from the sequence. More precisely,

$$\text{freq}(\alpha) = \max\{\text{freq(x)} : x \in \{1, \dots, \ell\}\}.$$

Note that, for all $1 \le i \le \ell$, $\text{freq}(i, \Psi^{(2,\ell)}) = 2$ and hence $\text{freq}(i, \Theta^\ell) = 2$. Thus $\text{freq}(\Psi^{(2,\ell)}) = 2$ and $\text{freq}(\Theta^\ell) = 2$. We will show some multicollision attacks on sequential hash functions based on sequences whose frequencies are at most two.

5 Multicollision Attacks on Generalized Sequential Hash Functions

For the sake of convenience, we slightly modify the notation used in the definition of the generalized sequential hash functions. Given a compression function $f : \{0,1\}^{n+n'} \to \{0,1\}^n$, a fixed initial value h_0, and a sequence $\alpha = \langle \alpha_1, \cdots, \alpha_s \rangle$ on symbols $\{1, \dots, \ell\}$, the *generalized sequential hash function based on* α is defined to be $H(m_1 \parallel \cdots \parallel m_\ell) = h_s$, where $h_i = f(h_{i-1}, m_{\alpha_i})$ for all $i \ge 1$.

We present a 2^r-way multicollision attack on the hash function based on a sequence α where $\text{maxchain}(\alpha) = r$. The complexity of the attack is $\text{O}(s\, 2^{n/2})$ where $s = |\alpha|$. In case of the classical iterated hash function, the corresponding sequence is $\Psi^{(1,\ell)}$ (see Example 4.1). Here we have $\text{maxchain}(\Psi^{(1,\ell)}) = \ell$ and thus we have a 2^ℓ-way multicollision attack with complexity $\text{O}(\ell\, 2^{n/2})$. This is the same as the complexity of Joux's attack. In fact, we will see that our attack is same as Joux's attack in the case of the classical iterated hash function.

The idea of the attack is to first identify message blocks that comprise a chain, and then to find a sequence of intermediate collisions by varying only those message blocks.

Example 5.1. We present an attack on the hash function based on the sequence

$$\Theta^5 = \langle 1, 2, 1, 3, 2, 4, 3, 5, 4, 5 \rangle$$

(see Example 4.3). Here $1 \prec 3 \prec 5$ is a chain in Θ^5. The attack proceeds as follows:

step 1 We first fix the message blocks m_2 and m_4 by defining their values to be equal to some arbitrary string IV.

step 2 We use a birthday attack to find $m_1^1 \neq m_1^2$ such that

$$f(f(f(h_0, m_1^1), IV), m_1^1) = f(f(f(h_0, m_1^2), IV), m_1^2) = h_3 \text{ for some } h_3.$$

step 3 We use a second birthday attack to find $m_3^1 \neq m_3^2$ such that

$$f(f(f(f(h_3, m_3^1), IV), IV), m_3^1) = f(f(f(f(h_3, m_3^2), IV), IV), m_3^2) = h_7 \text{ for some } h_7.$$

step 4 Finally, we use a third birthday attack to find $m_5^1 \neq m_5^2$ such that

$$f(f(f(h_7, m_5^1), IV), m_5^1) = f(f(f(h_7, m_5^2), IV), m_5^2) = h_{10} \text{ for some } h_{10}.$$

step 5 Define

$$\mathcal{M}_i = \begin{cases} \{m_i^1, m_i^2\} & \text{if } i \in \{1, 3, 5\} \\ \{IV\} & \text{if } i \in \{2, 4\}. \end{cases}$$

Then it is easy to see that

$$\mathcal{C} = \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3 \times \mathcal{M}_4 \times \mathcal{M}_5 \tag{2}$$

is a 2^3-way multicollision set with collision value h_{10}. The complexity of the attack is $10 \times 2^{n/2}$, because $|\Theta^5| = 10$.

Recall that the notation $h \xrightarrow{m} h'$ means $f(h, m) = h'$. More generally, given a sequence $m_1, m_2, \ldots, m_j$ we use the notation

$$h \xrightarrow{m_1, m_2, \ldots, m_j} h'$$

to mean

$$f \cdots (f(f(h, m_1), m_2), \ldots), m_j) = h'.$$

Then steps 2–4 of the attack can be rewritten using this notation, as follows:

$$\begin{array}{llll} h_0 \xrightarrow{m_1^1, IV, m_1^1} h_3 & \text{and} & h_0 \xrightarrow{m_1^2, IV, m_1^2} h_3 & \text{for some } h_3, \text{ where } m_1^1 \neq m_1^2 \\ h_3 \xrightarrow{m_3^1, IV, IV, m_3^1} h_7 & \text{and} & h_3 \xrightarrow{m_3^2, IV, IV, m_3^2} h_7 & \text{for some } h_7, \text{ where } m_3^1 \neq m_3^2 \\ h_7 \xrightarrow{m_5^1, IV, m_5^1} h_{10} & \text{and} & h_7 \xrightarrow{m_5^2, IV, m_5^2} h_{10} & \text{for some } h_{10}, \text{ where } m_5^1 \neq m_5^2. \end{array}$$

■

In general, we have the following multicollision attack on a generalized sequential hash function. As the proof idea is same as the example given above, we skip the proof. One can see the proof in [20].

Theorem 5.1. *Let H be a hash function based on a sequence α, where* $\text{maxchain}(\alpha) = r$. *Then there exists a 2^r-way multicollision attack on H with complexity $O(s\, 2^{n/2})$, where $s = |\alpha|$.*

Remark 5.1. Note that the above attack reduces to Joux's attack in the case of the classical iterated hash function. In this case, $1 \prec 2 \prec \cdots \prec \ell$ is a chain and thus we find a collision for each intermediate hash value by varying each message block. This is what Joux's attack does.

Remark 5.2. The attack illustrated in Example 5.1 is an application of Theorem 5.1. In this attack, we have $r = 3$ and the index set $\{1, \ldots, 10\}$ is partitioned into the three intervals $[1, 3]$, $[4, 7]$, and $[8, 10]$.

To get a 2^r-way collision on the hash function based on the sequence Θ^ℓ (see Example 4.3), we can take $\ell = 2r - 1$; then $\text{maxchain}(\Theta^\ell) = r$. By Theorem 5.1, we have a 2^r-way collision attack with complexity $\mathrm{O}(r\, 2^{n/2})$. However, we cannot apply the same idea to the hash function based on the sequence $\Psi^{(2,\ell)}$ because $\text{maxchain}(\Psi^{(2,\ell)}) = 1$ (see Example 4.2). Here, we have to use a different multicollision attack.

Example 5.2. We present an attack on the hash function based on the sequence $\Psi^{(2,\ell)}$. Here are the steps in the attack:

step 1 Define

$$t = \frac{n+1}{2} + \frac{\ln \ln 2r}{2 \ln 2}.$$

and let $\ell = tr$. We use Joux's multicollision attack on the hash function based on the sequence $\Psi^{(1,\ell)}$ to find ℓ pairs,

$$(m_1^1, m_1^2), (m_2^1, m_2^2), \cdots, (m_\ell^1, m_\ell^2),$$

such that

$$f(h_{i-1}, m_i^1) = f(h_{i-1}, m_i^2) = h_i,$$

$1 \le i \le \ell$. Thus we have a 2^ℓ-way collision set

$$\mathcal{C} = \{m_1^1, m_1^2\} \times \{m_2^1, m_2^2\} \times \cdots \times \{m_\ell^1, m_\ell^2\}$$

for the hash function based on the sequence $\Psi^{(1,\ell)}$.

step 2 Next, to get a 2^r-way multicollision for the hash function based on the sequence $\Psi^{(2,\ell)}$, we search for "intermediate" collisions within the set $\mathcal{C}$ using a standard birthday attack. Divide the index interval $[\ell+1, 2\ell]$ into r consecutive intervals, each consisting of t elements, i.e.,

$$\begin{aligned} I_1 &= [\ell+1, \ell+t], \\ I_2 &= [\ell+1+t, \ell+2t], \\ \vdots \;&\; \vdots \;\; \vdots \\ I_r &= [\ell+1+(r-1)t, \ell+rt]. \end{aligned}$$

Write $h'_0 = h_\ell$. Then, for each $1 \le i \le r$, try to find two t-tuples, say $M_i^1 \neq M_i^2$, from the set

$$\mathcal{C}_i = \{m^1_{(i-1)t+1}, m^2_{(i-1)t+1}\} \times \{m^1_{(i-1)t+2}, m^2_{(i-1)t+2}\} \times \cdots \times \{m^1_{it}, m^2_{it}\}$$

such that

$$h'_{i-1} \xrightarrow{M_i^1} h'_i \quad \text{and} \quad h'_{i-1} \xrightarrow{M_i^2} h'_i \quad \text{for some } h'_i,$$

say. If this step fails for any i, we have to return to step 1 and find a new 2^ℓ-way collision set $\mathcal{C}$.

step 3 Provided that the r birthday attacks in step 2 all succeed, it is easy to observe that

$$\mathcal{C}^* = \{M_1^1, M_1^2\} \times \{M_2^1, M_2^2\} \times \cdots \times \{M_r^1, M_r^2\}$$

is a multicollision set (of size 2^r) for our hash function[2].

[2] This is a slight abuse of notation. Each element in this cartesian product is an r-tuple of t-tuples, whereas what we really want is the corresponding rt-tuple.

We need to compute the success probability of this attack. Applying Theorem 2.1, each birthday attack in step 2 succeeds with probability

$$1 - e^{-(2^t)^2/2^{n+1}} = 1 - e^{-2^{2t-(n+1)}} = 1 - e^{-2^{\frac{\ln\ln 2r}{\ln 2}}} = 1 - \frac{1}{2r}.$$

The probability that all r birthday attacks in step 2 succeed is

$$\left(1 - \frac{1}{2r}\right)^r > \frac{1}{2}.$$

Now we can estimate the complexity of the entire attack. The expected complexity of step 1 is $2.5tr\,2^{n/2}$. The complexity of any given birthday attack in step 2 is analyzed as follows: We need two queries to f for the two values of the first message block in an interval. Then we need four queries for the two values of the next message block. Continuing for t message blocks, the complexity is computed to be at most[3]

$$2 + 4 + 8 + \cdots + 2^t = 2(2^t - 1).$$

Then, the complexity of the r birthday attacks in step 2 (assuming they all succeed) is at most $2r(2^t - 1)$. Since step 2 succeeds with probability 1/2, the overall expected complexity is upper-bounded by

$$2 \times (2.5tr\,2^{n/2} + 2r(2^t - 1)).$$

Using the fact that t is $\Theta(n + \ln\ln r)$ and 2^t is $\Theta(2^{n/2}\ln r)$, the overall expected complexity is $\mathrm{O}(r(n + \ln\ln r)2^{n/2} + r\ln r\,2^{n/2})$, which is $\mathrm{O}(r(n + \ln r)2^{n/2})$. ■

Now, we state a theorem which is a generalization of the attack given in Example 5.2. Once again we skip the proof and one can find the proof in![20].

Theorem 5.2. *Suppose that α is a sequence of length s on symbols $\{1, \ldots, \ell\}$ such that*

1. $\mathrm{freq}(\alpha) \le 2$, *and*
2. *there exists an initial interval $[1, w]$ of the index set for which $\alpha[1, w]$ contains tr symbols having frequency 1, where $t = (n+1)/2 + (\ln\ln 2r)/(2\ln 2)$.*

Then there is a 2^r-way collision attack on the hash function based on the sequence α having complexity $O(s\ \ln r\, 2^{n/2})$.

So far, we have provided a multicollision attack if the underlying sequence α satisfies certain conditions. Now we show that appropriate conditions hold for any sequence α (having a sufficient number of elements) provided that $\mathrm{freq}(\alpha) \le 2$.

Theorem 5.3. *Let α be a sequence of elements from symbol set $\mathcal{S} = \{1, \ldots, \ell\}$ such that $1 \le \mathrm{freq}(x, \alpha) \le 2$ for all $x \in \mathcal{S}$. Suppose that $\ell \ge uv$. Then one of the following holds:*

1. $\mathrm{maxchain}(\alpha) \ge u$, *or*

[3] This computation assumes that no collisions occur before the last message block in the given interval. If a collision occurs earlier, then the complexity is reduced.

2. *there exists an initial interval* $[1, w]$ *of the index set such such that* $\alpha[1, w]$ *contains at least* v *symbols each having frequency* 1*.*

Proof. Let $\text{maxchain}(\alpha) = u_0$. If $u_0 \geq u$, then we're done, so suppose that $u_0 < u$. Let v_0 denote the maximum number of mutually incomparable symbols in $\mathcal{S}$. By Dilworth's theorem, there exists a decomposition of $\mathcal{S}$ into v_0 chains. Every chain contains at most u_0 symbols, so $|\mathcal{S}| = \ell \leq u_0 v_0 < u v_0$. Hence, $v_0 > \ell/u \geq v$.

Now, let $x_{i_1}, \ldots, x_{i_v}$ be v mutually incomparable elements. Let the first occurrence of x_{i_j} be at position a_j of α, for $j = 1, \ldots, v$, where $a_1 < \cdots < a_v$. Recall that every symbol occurs at most twice in α. It then follows that the second occurrence (if any) of any x_{i_j} must be after position a_v of α (if not, then $x_{i_j} \prec x_{i_v}$, a contradiction). Therefore $x_{i_1}, \ldots, x_{i_v}$ all occur exactly once in the subsequence $\alpha[1, a_v]$. □

Now we state and prove our main theorem, which demonstrates a multicollision attack for any generalized sequential hash function with frequency at most two.

Theorem 5.4. *Let* H *be a generalized sequential hash function based on the sequences* $\langle \alpha^1, \alpha^2, \cdots \rangle$*, where* $\text{freq}(\alpha^\ell) \leq 2$ *for every* $\ell \geq 1$*. Then we have a* 2^r*-way collision attack on* H *having complexity* $O(r^2 \ln r\,(n + \ln\ln r) 2^{n/2})$*.*

Proof. Let $t = (n+1)/2 + (\ln\ln 2r)/(2\ln 2)$ and define $\ell = r^2 t$. Applying Theorem 5.3, we have that one of the following holds:

1. $\text{maxchain}(\alpha^\ell) \geq r$, or

2. there exists an initial interval $[1, w]$ of the index set such such that $\alpha[1, w]$ contains at least rt symbols each having frequency 1.

In the first case, Theorem 5.1 provides a 2^r-way collision attack having complexity $O(s\, 2^{n/2})$, where $s = |\alpha^\ell|$. But $s \leq 2\ell$, so s is $O(r^2(n + \ln\ln r))$ and the attack has complexity $O(r^2\,(n + \ln\ln r) 2^{n/2})$.

In the second case, we apply Theorem 5.2. Here, the attack has complexity $O(s \ln r\, 2^{n/2})$, which is $O(r^2 \ln r\,(n + \ln\ln r) 2^{n/2})$. This complexity exceeds the complexity of case 1, and the result follows. □

6 Conclusion

In this paper, we have defined a natural class of hash functions and we studied their security with respect to multicollision attacks. We have found efficient multicollision attacks on the generalized sequential hash functions when the message blocks are processed at most most twice. Hoch and Shamir [11] recently have found attacks even if the message blocks are used more than twice. It is an interesting open question to search for hash functions, within the general class we defined in Section 3, which are secure against multicollision attacks.

Acknowledgements

Research of the second author is supported by NSERC grant RGPIN 203114-02.

References

[1] M. Bellare and T. Kohno. Hash function balance and its impact on birthday attacks. *Lecture Notes in Computer Science* **3027** (2004), 401–418 (Eurocrypt 2004).

[2] E. Brickell, D. Pointcheval, S. Vaudenay and M. Yung. Design validations for discrete logarithm based signature schemes. *Lecture Notes in Computer Science* **1751** (2000), 276–292 (PKC 2000).

[3] D. Coppersmith. Another birthday attack. *Lecture Notes in Computer Science* **218** (1996), 14–17 (CRYPTO '85).

[4] I. B. Damgård. A design principle for hash functions. *Lecture Notes in Computer Science* **435** (1990), 416–427 (CRYPTO '89).

[5] D. W. Davies and W. L. Price. The application of digital signatures based on public key cryptosystems. In *Proceedings of the Fifth International Computer Communications Conference*, 1980, pp. 525–530.

[6] P. Diaconis and F. Mosteller. Methods for studying coincidences. *Journal of the American Statistical Association* **84** (1989), 853–861.

[7] M. Girault, R. Cohen and M. Campana. A generalized birthday attack. *Lecture Notes in Computer Science* **330** (1988), 129–157 (EUROCRYPT '88).

[8] M. Girault and J. Stern. On the length of cryptographic hash-values used in identification schemes. *Lecture Notes in Computer Science* **839** (1994), 202–215 (CRYPTO '94).

[9] M. Hattori, S. Hirose and S. Yoshida. Analysis of double block length hash functions. *Lecture Notes in Computer Science* **2898** (2003), 290–302 (Cryptography and Coding 2003).

[10] S. Hirose. Provably secure double-block-length hash functions in a black-box model. To appear in *Lecture Notes in Computer Science* (ICISC 2004).

[11] J. J. Hoch and A. Shamir. Finding multicollisions in iterated concatenated and expanded hash functions. Preprint.

[12] A. Joux. Multicollisions in iterated hash functions. Application to cascaded constructions. *Lecture Notes in Computer Science* **3152** (2004), 306–316 (CRYPTO 2004).

[13] J. Kelsey and B. Schneier. Second preimages on n-bit hash functions for much less than 2^n work. *IACR Cryptology ePrint Archive*, Report 2004/304, `http://eprint.iacr.org/2004/304`.

[14] L. Knudsen, X. Lai and B. Preneel. Attacks on fast double block length hash functions. *Journal of Cryptology* **11** (1998), 59–72.

[15] L. Knudsen and B. Preneel. Construction of secure and fast hash functions using nonbinary error-correcting codes. *IEEE Transactions on Information Theory* **48** (2002), 2524–2539.

[16] S. Lucks. Design principles for iterated hash functions. *IACR Cryptology ePrint Archive*, Report 2004/253, `http://eprint.iacr.org/2004/253`.

[17] A. J. Menezes, P. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1996.

[18] R. Merkle. One way hash functions and DES. *Lecture Notes in Computer Science* **435** (1990), 428–446 (CRYPTO '89) .

[19] M. Nandi. *Design of Iteration on Hash Functions and its Cryptanalysis.* PhD thesis, Indian Statistical Institute, 2005.

[20] M. Nandi and D. R. Stinson. Multicollision attacks on generalized hash functions. *IACR Cryptology ePrint Archive*, report 2004/330, `http://eprint.iacr.org/2004/330`.

[21] B. Preneel. *Analysis and Design of Cryptographic Hash Functions.* Doctoral Dissertation, Katholieke Universiteit Leuven, 1993.

[22] R. Rivest and A. Shamir. PayWord and MicroMint. *CryptoBytes* **2**(1) (1996), 7–11.

[23] P. Sarkar. Domain extender for collision resistant hash functions: improving upon Merkle-Damgård iteration. *IACR Cryptology ePrint Archive*, Report 2003/173, `http://eprint.iacr.org/2003/173`.

[24] T. Satoh, M. Haga and K. Kurosawa. Towards secure and fast hash functions. *IEICE Trans. Fundamentals,* **E82-A**, no. 1, January, 1999.

[25] D. R. Stinson. *Cryptography: Theory and Practice, Second Edition*, CRC Press, 2002.

[26] D. R. Stinson. Some observations on the theory of cryptographic hash functions. To appear in *Designs, Codes and Cryptography.*