Multicollision Attacks on a Class of Hash Functions

M. Nandi
Applied Statistics Unit
Indian Statistical Institute
Calcutta, India
mridul_r@isical.ac.in

D. R. Stinson
School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
dstinson@uwaterloo.ca

April 22, 2005

Abstract

In a recent paper, A. Joux [7] showed multicollision attacks on the classical iterated hash function. (A multicollision is a set of inputs whose hash values are same.) He also showed how the multicollision attacks can be used to get a collision attack on the concatenated hash function. In this paper, we first try to fix the attack by introducing a natural and wide class hash functions. However, we show that the multicollision attacks also exist in this general class. Thus, we rule out a natural and a wide class of hash functions as candidates for multicollision secure hash functions.

1 Introduction

In this paper, we discuss multicollision attacks on certain general classes of hash functions. A multicollision is a generalized notion of collision on a function. A *collision* on a function $g : D \to R$ is a doubleton subset $\{x, y\}$ of D such that $g(x) = g(y)$. For an integer $r \geq 2$, an *r-way collision* (or *multicollision*) on a function, $g(\cdot)$, is an r-subset $\{x_1, \ldots, x_r\}$ of D 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 multicollision attack is an algorithm which finds an r-way collision set with some probability.

The *birthday attack* for r-way collisions has time complexity $\emptyset(|R|^{(r-1)/r})$. In particular, the time complexity for finding a collision is $O(|R|^{1/2})$ (here, $r = 2$). 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* [3] [13], $H : \{0,1\}^* \to \{0,1\}^n$, is based on a *compression function*, $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. 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 m (see [3] [13] [12] 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 $M = m_1 \parallel \cdots \parallel m_l$ be a *padded message* with $|m_i| = m, 1 \leq i \leq l$ 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 $h_i = f(h_{i-1}, m_i)$, $1 \leq i \leq l$. For each $1 \leq i < l$, 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 [7] found an algorithm to construct a 2^K-way multicollision set on a classical iterated hash function, having time complexity $O(K\, 2^{n/2})$, which is a considerable improvement over the birthday attack. 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. This concatenated hash function has often been used when large output hash values are needed.

2. There are some other practical applications where multicollision secure hash functions are required. These include the micropayment scheme Micromint [15], the identification scheme of Girault and Stern [4], 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

In the light of the above discussion, it remains an open question how to find a good design technique for hash functions that are secure against multicollision attack. In this paper, we provide some negative results to this question. In particular, we study a natural generalization of the classical iterated hash function. This generalization is a natural way to attempt to fix the classical iterated design and it is therefore worthwhile to study this approach in detail. Unfortunately, we show that these generalized hash functions also have multicollision attacks: we find 2^K-way collision attacks with complexity $O(n\, K^2\, 2^{n/2})$. We also find multicollision attacks on a certain class of generalized tree-based hash functions, having complexity $O(n\, K^2\, 2^{n/2})$. Tree-based hash functions can be viewed as a parallellization of the classical iterated hash functions (which are evaluated in a sequential manner). Thus, we rule out a natural and large class of general hash functions as candidates for multicollision secure hash functions.

1.2 Organization of Our Paper

We first give a brief introduction of Joux's attack and its applications. Then we generalize the definition of classical iterated hash functions in a natural way. After defining this class of generalized iterated hash functions, we show some multicollision attacks on them. We also provide multicollision attacks for tree-based hash functions. Finally, we conclude with a brief note on multipreimage attacks and possible future work.

2 Preliminaries and Related Works

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 attack.

2.1 The Birthday Attack

A hash function usually has two main components: a compression function, $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, 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 output 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 K-way collision attack algorithm is the number of queries of f required to get a K-way collision. A natural (and the most popular) attack is the *birthday attack*. The following algorithm describes the birthday attack for (possibly) finding an r-way collision of the function g, where the attack has complexity q.

BirthdayAttack(g, q, r) **:**

1. Choose $x_1, \cdots, x_q$ randomly from the domain D and compute $y_i = g(x_i)$ for $1 \le i \le q$.
2. Return a subset (if any) $C \subseteq \{x_1, \cdots, x_q\}$ of size r such that C is an r-way multicollision subset for the function g. Otherwise return the output "failure".

The next proposition gives an estimate of the complexity of the birthday attack in finding an r-way collision with significant probability. Here we assume that the function, $g : D \to R$, is a *random oracle*. (A function g is a random oracle if every value $g(x)$ is chosen uniformly at random from R for every $x \in D$.) Random oracles are the usual model for hash functions which can be accessed a black-box manner. See [1, 12, 18, 19] for a more detailed discussion of the standard birthday attack for finding 2-way collisions.

Proposition 1 (Complexity of the Birthday Attack)
For a random oracle $g : D \to \{0,1\}^n$, the birthday attack with complexity q finds an r-way collision with probability $O(q^r/2^{(r-1)n})$. Thus, the birthday attack requires $\Omega(2^{n(r-1)/r})$ queries to find an r-way collision with significant probability. For $r = 2$, the (standard) birthday attack requires $\Omega(2^{n/2})$ queries.

Proof. Since $x_1, \cdots, x_q$ are randomly chosen from the domain D, $g(x_1), \cdots, g(x_q)$ are independent and uniformly distributed over R. Thus, for any $\{i_1, \cdots, i_r\} \subset [1, q]$ we have,

$$\Pr[g(x_{i_1}) = \cdots = g(x_{i_r})] = 1/2^{(r-1)n}.$$

Let $C_1, \cdots, C_k$ be r-subsets of $\{x_1, \cdots, x_q\}$ (which denotes the complete query list of the birthday attack) where $k = \binom{q}{r}$. Let E_i denote the event that C_i is a r-way multicollision set. Thus, the event corresponding to the existence of an r-way collision in the set $\{x_1, \cdots, x_q\}$ is $\bigcup_i E_i$. Hence, the probability that the birthday attack finds an r-way collision set is

$$\Pr(\textstyle\bigcup_i E_i) \le \sum_i \Pr[E_i] = \frac{\binom{q}{r}}{2^{(r-1)n}} = O(q^r/2^{(r-1)n}).$$

Thus, the birthday attack requires $\Omega(2^{n(r-1)/r})$ queries to find an r-way collision with significant probability. ■

2.2 Joux's Multicollision Attack

In a recent paper by Joux [7], it was shown that there is a 2^r-way collision attack for the classical iterated hash function based on a compression function, $f : \{0,1\}^{m+n} \to \{0,1\}^n$, where the attack has complexity $O(r\, 2^{n/2})$. This complexity is much less than $\Omega(2^{\frac{n(2^r-1)}{2^r}})$, which is the complexity for the birthday attack (see Proposition 1).

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

$$\begin{aligned} f(h_0, m_1) = f(h_0, n_1) &= h_1 \text{ (say), where } m_1 \neq n_1 \\ f(h_1, m_2) = f(h_1, n_2) &= h_2 \text{ (say), where } m_2 \neq n_2 \\ &\vdots \\ f(h_{r-1}, m_r) = f(h_{r-1}, n_r) &= h_r \text{ (say), where } m_r \neq n_r. \end{aligned}$$

For $1 \leq i \leq r$, we apply BirthdayAttack$(f(h_{i-1}, \cdot), 2^{n/2}, 2)$ to find $m_i \neq n_i$ such that $f(h_{i-1}, m_i) = f(h_{i-1}, n_i)$. Thus the set

$$\{x_1 \parallel \cdots \parallel x_r : x_i = m_i \text{ or } n_i, 1 \leq i \leq r\}$$

is a 2^r-way collision set. The complexity of the attack is $O(r\, 2^{n/2})$.

Figure 1 is a diagram illustrating the attack.

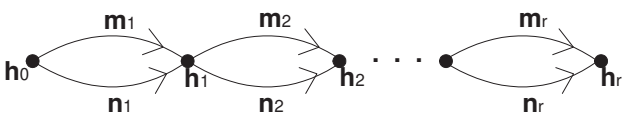

Figure 1: Graphical representation of Joux's multicollision attack

Remark: Since the birthday attack is itself a probabilistic attack, there is some positive probability that Joux's attack fails. One can perform the birthday attack repeatedly until each required intermediate collision is found. If ϵ is the success probability of the birthday attack, then on the average we need r/ϵ calls of BirthdayAttack in total. In fact, if we make a total of $2r/\epsilon$ calls of BirthdayAttack, then we have an overall success probability exceeding 1/2. This is because of the following fact which follows immediately from Markov's Inequality: if the expected value of a random variable, X, is μ, then $\Pr[X \leq 2\mu] \geq 1/2$.

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 [5], [6], [10], [9], [14], [17]. However, most of these designs were proven to be insecure; see [5], [10] [9], [14], [17]. Recently, Hirose [6] 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 [7], 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 [7] 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 $O(n_G\, 2^{n_H/2} + 2^{n_G/2})$. Note that we only assume that H is a classical iterated hash function; G can be any hash function at all.

Remark: 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, very recently, Kelsey and Schneier [8] 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 [11] 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 [16]. One can also combine these two approaches.

These generalized hash functions could be considered as an alternative to the classical iterated hash function since the 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 by *double iterated hash function* as it uses the classical iteration 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.

To begin with, we define a very general class of hash functions. Let $f : \{0,1\}^N \to \{0,1\}^n$ 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 $M = m_1 \parallel \cdots \parallel m_l$, where each m_i is a message block that is an m-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 strings. Every input to f is of the form $h \parallel x$. Each h is the concatenation of r n-bit strings, each of which is a previously computed output of f or an initial value; and each x is the concatenation of q message blocks. We will require that $r \geq 0$, $q \geq 1$ and $nr + mq = N$.

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:

$$\begin{array}{l} f(h_i \parallel x_i) = y_i, \\ h_i = h_i^1 \parallel \cdots \parallel h_i^r, \\ h_i^j \in \{v_1, v_2, \cdots\} \cup \{y_1, \cdots, y_{i-1}\}, \\ x_i = x_i^1 \parallel \cdots \parallel x_i^q, \quad \text{and} \\ x_i^j \in \{m_1, \cdots, m_l\}. \end{array}$$

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

In this paper, we consider two special (but still quite general) subclasses of $\mathcal{D}$. They are termed *generalized sequential hash functions* (denoted $\mathcal{S}$) and *generalized binary tree based hash functions* (denoted $\mathcal{T}$). We show multicollision attacks on certain subclasses of $\mathcal{S}$ and $\mathcal{T}$.

Generalized Sequential Hash Functions

In the class $\mathcal{S}$ of generalized sequential hash functions, we have $r = q = 1$ for all i and $N = m + n$. Define a sequence $\alpha = \langle \alpha_1, \cdots, \alpha_s \rangle$ where $\alpha_i \in [1, l] = \{1, 2, \ldots, l\}$. 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

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

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:

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

Note that a message block can be used more than once. In the case of the classical iterated hash function, however, we have $\alpha_i = i$ for all i, and $s = l$. Also, in the classical iterated hash function, each message block is used exactly once.

To define a hash function with arbitrary domain $(\{0,1\}^m)^*$, say $H : (\{0,1\}^m)^* \rightarrow \{0,1\}^n$, 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 $[1, l]$, whenever M is an l-block message.

Generalized Tree based Hash Functions

We also consider a class $\mathcal{T}$ of binary tree based hash functions in this section. We assume that $N = 2n$. so the compression function $f : \{0,1\}^{2n} \rightarrow \{0,1\}^n$.

We can specify the computation of the hash function by a list of triples

$$L = \{(z_i^1, z_i^2, y_i) : 1 \leq i \leq s\},$$

where the following conditions hold:

$$f(z_i^1 \parallel z_i^2) = y_i, \quad \text{and}$$
$$z_i^1, z_i^2 \in \{v_1, v_2, \cdots\} \cup \{y_1, \cdots, y_{i-1}\} \cup \{m_1, \cdots, m_l\}.$$

We describe this class in more detail in Section 5.

4 Attacks on Generalized Sequential Hash Functions

4.1 Some Terminologies on Sequences

Consider a finite sequence $\alpha = \langle \alpha_1, \alpha_2, \cdots, \alpha_s \rangle$ of $[1, l]$. The *length* of the sequence is s and it is denoted by $|\alpha|$. The *index set* of the sequence is $[1, s]$. Given any subset $I = \{i_1, \cdots, i_t\}$ of the index set $[1, s]$, we have a *subsequence* $\alpha(I) = \langle \alpha_{i_1}, \cdots, \alpha_{i_t} \rangle$ where, $i_1 < \cdots < i_t$. Sometimes, we use the same notation $\alpha(I)$ to denote the multiset $\{\alpha_{i_1}, \cdots, \alpha_{i_t}\}$.

We say *subintervals* $I_1, \cdots, I_d$ form a *partition* of $I \subseteq [1, s]$ if there exists

$$a_1 = 1 < b_1 + 1 = a_2 < b_2 + 1 \cdots b_{d-1} + 1 = a_d < b_d = s$$

such that $[a_i, b_i] \cap I = I_i$. The I_i's are disjoint subintervals of I and their union is the whole set I. Let $J = [a, b] \cap I$ and suppose that the minimum elements of I and J are the same. Then we say that J is a *left-end subinterval* of I.

Definition 1 (Independent Elements in a Subsequence)
Distinct elements $x_1, \ldots, x_d \in [1, l]$ are said to be independent *in the subsequence $\alpha(I)$ if there exist d subsets $I_1, \ldots, I_d$ which form a partition of I such that $x_i \in \alpha(I_i)$ but $x_i \notin \alpha(I_j)$ for $1 \leq j \neq i \leq d$.*

Observation: $x_1, \cdots, x_d$ are independent in α if and only if any subsequence containing $x_1, \cdots, x_d$ of α is of the form

$$\langle x_1, \cdots, x_1, x_2, \cdots, x_2, \cdots, x_d, \cdots, x_d \rangle.$$

$x_1, \cdots, x_t$ cannot be independent in α if there is a subsequence $\langle x_i, x_j, x_i \rangle$ of α for some $1 \leq i \neq j \leq t$; this is because any interval containing all occurrences of x_i also contains x_j.

We write

$$\mathcal{I}(\alpha(I)) := \max\{d : \exists\ d \text{ independent elements in the subsequence } \alpha(I)\}.$$

Thus, existence of d independent elements in $\alpha(I)$ implies $\mathcal{I}(\alpha(I)) \geq d$. Obviously, for any subsequence α_1, it holds that $\mathcal{I}(\alpha_1) \geq 1$. If there are k elements which appear only once in the sequence α, then $\mathcal{I}(\alpha) \geq k$.

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 1 *Let $\vartheta^{(1)} = \langle 1, 2, \ldots, l \rangle$ (the sequence for the classical iterated hash function). It is easy to note that $\mathcal{I}(\vartheta^{(1)}) = l$.*

Example 2 *Let $\vartheta^{(2)} = \langle 1, 2, \ldots, l, 1, 2, \ldots, l \rangle$. The* doubly iterated hash function *is based on the sequence $\vartheta^{(2)}$. It is easy to observe that there are no two independent elements in the sequence $\vartheta^{(2)}$ and hence $\mathcal{I}(\vartheta^{(2)}) = 1$. But, $\mathcal{I}(\vartheta^{(2)}([1, l])) = \mathcal{I}(\vartheta^{(1)}) = l$.*

Example 3 *Let $\theta^l = \langle 1, 2, 1, 3, 2, 4, 3, \cdots, l-1, l-2, l, l-1, l \rangle$. Here, $\mathcal{I}(\theta^l) \geq \lfloor \frac{l+1}{2} \rfloor$ as $1, 3, \cdots, l$ (if l is odd) or $1, 3, \cdots, l-1$ (if l is even) are independent elements. To show this, consider the partition $[1, 3], [4, 7], [8, 11], \cdots [2l-2, 2l]$ of $[1, 2l]$. Assume that l is odd (the other case can be proved similarly). Now one can see that $1 \in \theta^l[1, 3]$, $3 \in \theta^l[4, 7]$, $\cdots$, $l \in \theta^l[2l-2, 2l]$. These elements are not appearing in other intervals. Thus $1, 3, \cdots, l$ are independent elements (see Definition 1) for the sequence θ^l.*

In fact, we have $\mathcal{I}(\theta^l) = \lfloor \frac{l+1}{2} \rfloor$. For any $\lfloor \frac{l+1}{2} \rfloor + 1$ elements from $[1, l]$ there are two consecutive elements (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 $\lfloor \frac{l+1}{2} \rfloor + 1$ elements can be independent (see the observation above).

For a sequence α of $[1, l]$ and $x \in [1, l]$ 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,

$$\text{freq}(\alpha) = \max\{\text{freq(x)} : x \in [1, l]\}.$$

Note that, for all $1 \leq i \leq l$, $\text{freq}(i, \vartheta^{(2)}) = 2$ and hence $\text{freq}(i, \theta^l) = 2$. Thus $\text{freq}(\vartheta^{(2)}) = 2$ and $\text{freq}(\theta^l) = 2$. We show some multicollision attacks on sequential hash functions based on sequences whose *frequencies* are at most two.

4.2 Multicollision Attacks on Generalized Sequential Hash Function

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

We present a 2^r-way multicollision attack on the hash function based on a sequence α, where $\mathcal{I}(\alpha) = r$. The complexity of the attack is $O(s\, 2^{n/2})$ where $s = |\alpha|$. In case of the classical iterated hash function, the corresponding sequence is $\vartheta^{(1)}$ (see Example 1). Here we have $\mathcal{I}(\vartheta^{(1)}) = l$ and thus we have a 2^l-way multicollision attack with complexity $O(l\, 2^{n/2})$. 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 some independent message blocks, and then to find a sequence of intermediate collisions by varying only those message blocks.

First, we illustrate our attack for the hash function based on the sequence $\theta^5 = \langle 1,2,1,3,2,4,3,5,4,5 \rangle$ (see Example 3). Here $1,3,5$ are independent elements in θ^5. The attack proceeds as follows:

1. We first fix the message blocks m_2, m_4 and m_6 arbitrarily, by defining their values to be equal to some string IV.

2. Then we find $m_1^1 \neq m_1^2$ such that

$$f(f(f(h_0, m_1^1), IV), m_1^1) = f(f(f(h_0, m_1^2), IV), m_1^2) = h_1.$$

3. Then, we find find $m_3^1 \neq m_3^2$ such that

$$f(f(f(f(h_1, m_3^1), IV), IV) m_3^1) = f(f(f(f(h_1, m_3^2), IV), IV) m_3^2) = h_2.$$

4. Finally, we find $m_5^1 \neq m_5^2$ such that

$$f(f(f(h_2, m_5^1), IV), m_5^1) = f(f(f(h_2, m_5^2), IV), m_5^2) = h_3.$$

5. Now it is easy to see that

$$C = \{m : m_i = m_i^1 \text{ or } m_i^2 \text{ for } i = 1,3,5, m_i = IV \text{ for } i = 2,4,6\} \tag{1}$$

is a 2^3-way multicollision set with collision value h_3 (see Figure 2). The complexity of the attack is $O(10 \times 2^{n/2})$, because $|\theta^5| = 10$.

The attack is depicted diagramatically in Figure 2.

Before giving the proof of a general attack based on the ideas used in this example, we first want to introduce some additional notation and rewrite the above attack in terms of this notation. Referring to the previous attack, we introduce some notation relating to h_1, h_2 and h_3. These are the intermediate hash values where we are looking for intermediate collisions.

Consider the algorithm for computing a generalized sequential hash function. In the ith round, we have $h_i = f(h_{i-1}, m_{\alpha_i})$. We can define certain partial computations in that algorithm. Define $H(h^*, [a,b], M) = h_b$ when computing $H(M)$, given that $h_a = h^*$. Note that h_a and h_b are the

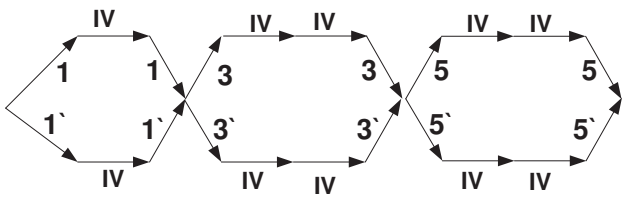

Figure 2: Graphical representation of Multicollision attack on the hash function based on the sequence θ^5

intermediate hash values at rounds a and b, respectively. So, $H(h^*, [a, b], M)$ is the hash value at round b given that we start with the intermediate hash value h^* at round a.

Now, we rewrite the above-described attack in terms of the notation just defined. We have the partition $[1, 3], [4, 7]$ and $[8, 10]$ and independent elements 1, 3 and 5 (see Example 3). We have fixed the message blocks m_2, m_4 and m_6 to be equal to a certain string IV. Thus $H(h_0, [1, 3], M)$ depends on only on the message block m_1 and hence we can write $H(h_0, [1, 3], m_1)$ instead of $H(h_0, [1, 3], M)$. Similarly, we write $H(h_1, [4, 7], m_3)$ and $H(h_2, [8, 10], m_5)$ instead of $H(h_1, [4, 7], M)$ and $H(h_2, [8, 10], M)$ respectively.

Then the 2^3-way collision attack can be described succinctly as follows :

1. Find $m_1^1 \neq m_1^2$ such that $H(h_0, [1, 3], m_1^1) = H(h_0, [1, 3], m_1^2) = h_1$ (say).
2. Find $m_2^1 \neq m_2^2$ such that $H(h_1, [4, 7], m_3^1) = H(h_1, [4, 7], m_3^2) = h_2$ (say).
3. Find $m_3^1 \neq m_3^2$ such that $H(h_2, [8, 10], m_5^1) = H(h_2, [8, 10], m_5^2) = h_3$.
4. Then C (as defined in (1)) is a 2^3-way multicollision set with collision value h_3.

In general, we have the following multicollision attack on a generalized sequential hash function.

Proposition 2 *Let H be a hash function based on a sequence $\alpha = \alpha^l$, where $\mathcal{I}(\alpha) = r$. Then we have a 2^r-way multicollision attack on H with complexity $\mathrm{O}(s\, 2^{n/2})$, where $s = |\alpha|$.*

Proof. As $\mathcal{I}(\alpha) = r$, we have r independent elements $x_1, \ldots, x_r$ and r disjoint and exhaustive subintervals, $I_1, \cdots, I_r$. Now fix the message blocks m_i to be equal to an arbitrary string IV, for all $i \notin \{x_1, \ldots, x_r\}$.

Because the x_i's are independent, it follows that $H(h^*, I_i, M)$ will only depend on m_{x_i} for all i. Thus, for simplicity, we write $H(h^*, I_i, m_{x_i})$ instead of $H(h^*, I_i, M)$.

Now find r successive collisions as follows:

$$H(h_{i-1}, I_i, m_{x_i}^1) = H(h_{i-1}, I_i, m_{x_i}^2) = h_i,$$

for $1 \leq i \leq r$. Then, it is easy to check that,

$$\mathcal{C} = \{m_1 \parallel \ldots \parallel m_l : m_{x_1} = m_{x_1}^1 \text{ or } m_{x_1}^2, \ldots, m_{x_r} = m_{x_r}^1 \text{ or } m_{x_r}^2, m_i = IV \ \forall i \notin \{x_1, \ldots, x_r\}\}$$

is a multicollision set of size 2^r.

To get the ith intermediate collision, we need to make $\mathrm{O}(|\alpha(I_i)|\, 2^{n/2})$ queries of f. For the complete attack we need $\mathrm{O}(\sum_i |\alpha(I_i)|\, 2^{n/2}) = \mathrm{O}(|\alpha|\, 2^{n/2})$ queries of f. ∎

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

To get a 2^r-way collision on the hash function based on the sequence θ^l (see Example 3), we can take $l = 2r - 1$. So $\mathcal{I}(\theta^l) = r$. By Proposition 2, 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 $\vartheta^{(2)}$ (since $\mathcal{I}(\vartheta^{(2)}) = 1$; see Example 2). Here, we have to use a different multicollision attack. This attack is summarized as follows.

1. First, we take $l = rn/2$ and we use Joux's multicollision attack to find l pairs,

   $$(m_1^1, m_1^2), (m_2^1, m_2^2), \cdots, (m_l^1, m_l^2),$$

   such that

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

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

   $$\mathcal{C} = \{m : m_i = m_i^1 \text{ or } m_i = m_i^2\}$$

   for the hash function based on the sequence $\vartheta^{(2)}[1, l]$.

2. Next, to get a 2^r-way multicollision for the desired hash function, we search for intermediate collisions within the set $\mathcal{C}$. Divide the index interval $[l+1, 2l]$ into r consecutive intervals, each consisting of $n/2$ elements, i.e., $I_1 = [l+1, l+n/2], \cdots, I_r = [l+1+(r-1)n/2, l+rn/2]$. Write $h'_0 = h_l$.

   Then, for each $1 \le i \le r$, find a pair $M_i^1 \ne M_i^2$ from the set

   $$\mathcal{C}_i = \left\{ m_{(i-1)n/2+1}^{j_1} \parallel \cdots \parallel m_{in/2}^{j_{n/2}} : j_1, \cdots j_{n/2} \in \{1, 2\} \right\}$$

   such that $H(h'_{i-1}, I_i, M_i^1) = H(h'_{i-1}, I_i, M_i^2) = h'_i$. Note that $|\mathcal{C}_i| = 2^{n/2}$ so the desired pair should exist.

3. Finally, it is easy to observe that

   $$\mathcal{C}^* = \{M_1^{j_1} \parallel \cdots \parallel M_r^{j_r} : j_1, \cdots j_r \in \{1, 2\}\}$$

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

See Figure 3 for a diagrammatic representation of the attack.

Now, we prove a general result which says that for any sequence α with frequency at most two with $\mathcal{I}(\alpha) \ge rn/2$ there is a 2^r-way collision attack with complexity $\mathrm{O}(r^2\, n\, 2^{n/2})$.

Proposition 3 *Let H be a hash function based on α^l with* $\mathrm{freq}(\alpha^l) \le 2$. *If there is a left-end subinterval I such that $\mathcal{I}(\alpha^l(I)) \ge rn/2$, then there is a 2^r-way multicollision attack on H having complexity* $\mathrm{O}(r^2\, n\, 2^{n/2})$.

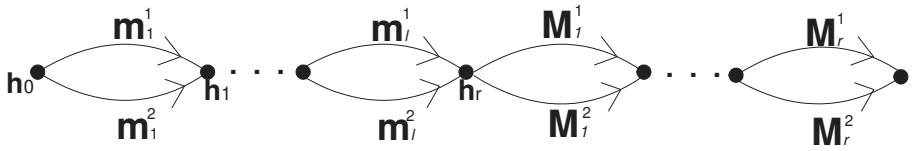

Figure 3: Graphical representation of multicollision attack on the hash function based on the sequence $\vartheta^{(2)}$

Proof. Let $x_1, \cdots, x_k$ be independent elements in $\alpha(I)$, where $k = rn/2$. As in Proposition 2, we have a set

$$\mathcal{C} = \{M = m_1 \parallel \ldots \parallel m_l; m_{x_1} = m_{x_1}^1 \text{ or } m_{x_1}^2, \ldots, m_{x_k} = m_{x_k}^1 \text{ or } m_{x_k}^2\}$$

of size 2^k so that $\mathcal{C}$ is a multicollision set for the hash function based on the sequence $\alpha(I)$. Let h_0' be the collision value for the multicollision set $\mathcal{C}$. Without loss of generality, we assume that each x_i appears exactly once in the sequence $\alpha([a+1, s])$, in the same order as they appear in I, where $I = [1, a]$ and s is the length of the sequence. (If this is not the case, then the proof can be modified suitably.)

Define $\mathcal{C}_{i+1}$ for $0 \leq i \leq r-1$ as follows:

$$\mathcal{C}_{i+1} = \left\{ m_{x_{in/2+1}}^{j_1} \parallel \cdots \parallel m_{x_{(i+1)n/2}}^{j_{n/2}} : j_1, \cdots, j_{n/2} \in \{1,2\} \right\}. \tag{2}$$

Now divide the interval $[a+1, s]$ into r disjoint and exhaustive subintervals $I_1', I_2', \cdots, I_r'$ so that $x_{in/2+1}, \cdots, x_{(i+1)n/2}$ appear in $I_{i+1}', 0 \leq i \leq r-1$. To make the notation simpler, we ignore all other message blocks as these are fixed to be equal to a string IV. We write

$$H(h^*, I_{i+1}', m_{x_{in/2+1}} \parallel \cdots \parallel m_{x_{(i+1)n/2}})$$

instead of

$$H(h^*, I_{i+1}', M).$$

Note that $|\mathcal{C}_i| = 2^{n/2}$. Then we find r successive collisions:

$$H(h_{i-1}', I_i', M_i^1) = H(h_{i-1}', I_i', M_i^2) = h_i',$$

for $1 \leq i \leq r$, where $M_i^1, M_i^2 \in \mathcal{C}_i$. Now it is easy to observe that

$$\mathcal{C}^* = \{M_1^{j_1} \parallel \cdots \parallel M_r^{j_r} : j_1, \cdots j_r \in \{1,2\}\}$$

is a multicollision set of size 2^r. ∎

So far, we have provided a multicollision attack if the underlying sequence satisfies certain conditions. More precisely, if $\mathcal{I}(\alpha) = r$ or if there exists an interval I such that $\mathcal{I}(\alpha(I)) = rn/2$, then there is a 2^r-way multicollision attack. Now we show that these conditions are satisfied by any sequence with a sufficient number of elements and having frequency at most two.

Definition 2 *Given any subsequence $\alpha(I)$ of α, we define*

$$S(\alpha(I)) = |\{x \in [1, l] : \text{freq}(x, \alpha(I)) \geq 1\}|.$$

Similarly, we can define

$$S^i(\alpha(I)) = |\{x \in [1, l] : \text{freq}(x, \alpha(I)) = i\}|.$$

So, when $\text{freq}(\alpha) \leq 2$ *we have* $S(\alpha(i)) = S^1(\alpha(i)) + S^2(\alpha(i))$.

Proposition 4 *Let α be a sequence of elements from $[1,l]$ with* $\mathrm{freq}(\alpha) \leq 2$ *and $S(\alpha) = l$. Suppose that $l \geq MN$. Then one of the folllowing holds:*

1. *$\mathcal{I}(\alpha) \geq M$, or*
2. *there exists a left-end subinterval I such that $\mathcal{I}(\alpha(I)) \geq N$.*

Proof. The proof is by induction on l. Let $|\alpha| = s$. For the left-end subinterval, $I = [1, N]$, either $\mathcal{I}(\alpha(I)) \geq S^1(\alpha(I)) = N$ or there exists an element, say x_1, which appears twice in the sequence $\alpha(I)$. In the former case we are done, so assume the latter. Remove all elements from α which appear in $\alpha(I)$ and call this new sequence $\alpha_1 = \alpha(I_1)$ for some set I_1.

Note that $S(\alpha_1) \geq MN - N = (M-1)N$. By induction, either $\mathcal{I}(\alpha_1) \geq M - 1$ or there exists a left-end subinterval J of I_1, such that $\mathcal{I}(\alpha_1(J)) \geq N$. In the latter case, $\mathcal{I}(\alpha([1,r])) \geq N$, where r is the last element in the set J. In this case, we are done. In the former case there exist $M-1$ independent elements $x_2, \ldots, x_M$ in the subsequence α_1. Also x_1 does not appear in the subsequence $\alpha[N+1, s]$ and $x_2, \ldots, x_M$ do not appear in $\alpha([1, N])$. Thus, $x_1, x_2, \ldots, x_M$ are independent elements in α. ■

Now we have a multicollision attack for any generalized sequential hash function with frequency at most two. This is immediate from Propositions 2, 3 and 4.

Theorem 5 *Let H be a generalized sequential hash function based on the sequences $\langle \alpha^1, \alpha^2, \cdots \rangle$, where* $\mathrm{freq}(\alpha^l) \leq 2$ *for every $l \geq 1$. Then we have a 2^r-way multicollision attack on H with complexity* $\mathrm{O}(r^2\, n\, 2^{n/2})$.

5 Multicollision attacks on generalized tree-based hash functions

Similar attacks can be carried out on generalized tree based hash functions. First, we define the generalized tree based hash function and some terminology. We modify the notation somewhat to make the attacks easier to describe.

Here we consider a compression function, $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, on which an l-block generalized tree based hash function $H(\cdot)$ is defined. Suppose that $m = m_1 \parallel m_2 \parallel \cdots \parallel m_l$ is an l-block message where every block is a bitsting of length n. Also, suppose that $h_1, h_2, \cdots \in \{0,1\}^n$ are constants (i.e., fixed initial values which only depend on l).

Define a list of s ordered pairs $\{(x_j^1, x_j^2)\}_{1 \leq j \leq s}$. For $1 \leq j \leq s$, we have that

$$x_j^1, x_j^2 \in \{h_1, h_2, \ldots\} \cup \{m_1, m_2, \cdots, m_l\} \cup \{z_1, \ldots, z_{j-1}\}$$

and $z_j = f(x_j^1, x_j^2)$. For $j \neq s$, the z_j's are the *intermediate hash values* and z_s is known as the final hash value. Finally, define $H(m) = z_s$.

We can assume that each intermediate hash value z_i and each message block m_j are in the list and hence they are inputs to some invocations of f. So there are no message blocks and intermediate hash values which are not hashed.

The above hash function also can be defined using a directed binary tree and a MIV (*message-initial value*) assignment. We first define the directed binary tree and some terminologies.

Definition 3 *A* directed binary tree *is a directed tree so that each vertex has indegree equal to either two or zero and outdegree equal to one, except for a unique vertex called the* root *which has outdegree equal to zero. A* leaf node *is a vertex with* indegree *zero. All other vertices or nodes*

(except the root) are known as intermediate nodes. *So intermediate nodes have indegree equal to two and outdegree equal to one. For every nonleaf node v, there are two nodes u and u' such that $(u,v),(u',v) \in E$. One of u, u' is denoted the* left child *of v and the other is denoted the* right child *of v.*

Now we state some terminology relating to directed binary trees.

1. Let $G = (V, E)$ be a rooted directed tree with root $q \in V$ and the arc set $E \subseteq V \times V$. We write $u \to v$ for the arc $(u, v) \in E$. Further, we write $u \Rightarrow v$ if there is a directed path from the vertex u to v or if $u = v$.

2. For a vertex v, define the subtree $G[v] = (V[v], E[v])$ to be the *subtree of G rooted at v*. That is, $V[v] = \{u \in V : u \Rightarrow v\}$ and $E[v] = \{(u, u') \in E : u, u' \in V[v]\}$.

3. We use the notation $L[G]$ (or simply L) for the set of leaves of G. Further, let $L[v]$ denote the set $L[G[v]]$, which is the set of leaves of the graph $G[v]$. Note that $L[v] = L \cap V[v]$.

Generalized Tree based Hash Functions

Let $G = (V, E)$ be a directed binary tree. An *MIV assignment* is a mapping $\rho : L \to [1, l] \cup \{0,1\}^n$. If $\rho(v) \in [1, l]$ then it denotes an index of a message block. When $\rho(v) \in \{0,1\}^n$, it denotes an initial value. Given a pair (G, ρ), we define a hash function H based on (G, ρ) as follows: For an l-block message $m = m_1 \| \cdots \| m_l$, we assign recursively an n-bit string to each vertex of G in the following manner:

1. For each leaf v, assign the n-bit string m_i to v if $\rho(v) = i$, or assign h to v if $\rho(v) = h$.

2. For any other node v, assign the n-bit string $f(z, z')$ to v, where z and z' are assigned to the vertices u and u' respectively, u is the left child of v and u' is the right child of v.

3. The output of the hash function, $H(m)$, is the value assigned to the root of the tree.

Now we define some more notation which will be used in the multicollision attack.

- For $x \in [1, l]$, we write $\mathrm{freq}(x, G)$ (or simply $\mathrm{freq}(x)$) for the number of times x appears in the multi-set $\rho(L)$ (this is called the *frequency* of x). That is, $\mathrm{freq}(x)$ denotes the number of times the message block m_x is hashed to get the final output hash value. Also, define $\mathrm{freq}(G) = \max\{\mathrm{freq}(x) : x \in L\}$.

- We define the *hash output at vertex v* (i.e., the value assigned to v, given that the message is m) to be $H(v, m)$. Note that a message block m_i is used to compute $H(v, m)$ if and only if i is in $\rho(L[v])$. We also use the notation $H(v, m_i)$ instead of $H(v, m)$ when $H(v, m)$ only depends on the ith message block, i.e., if the only index appearing in $\rho(L[v])$ is i.

- Given any vertex v, define $S(v, G)$ (or more simply, $S(v)$) to be the quantity

$$|\{x \in [1, l] : \mathrm{freq}(x, G[v]) \geq 1\}|.$$

  Similarly, we define

$$S^i(v) = |\{x \in [1, l] : \mathrm{freq}(x, G[v]) = i\}|.$$

  So $S(v)$ ($S^i(v)$, resp.) denotes the number of message blocks which are hashed at least once (exactly i times, resp.) to compute $H(v, m)$.

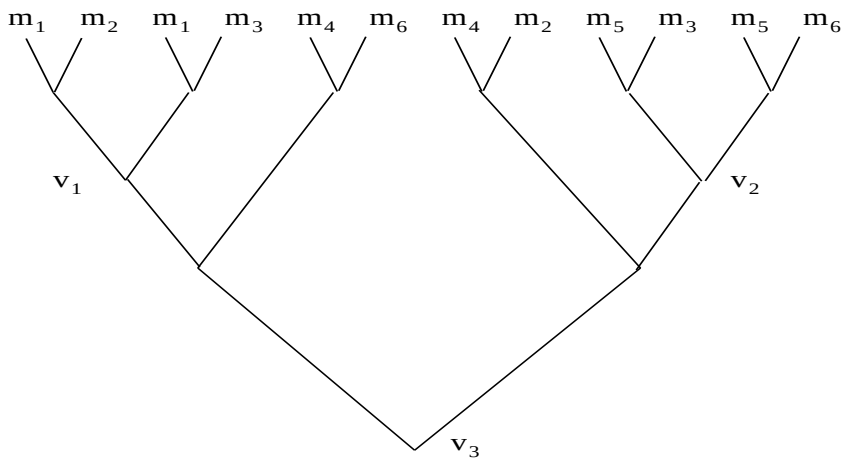

Figure 4: An example of 6-block binary tree based hash function.

Definition 4 (independent sequence of message indices) *Given a directed binary tree (G, ρ), we say that $(x_1, x_2, \ldots, x_k)$ is an* independent sequence *of message indices if there exist vertices $v_1, v_2, \ldots, v_k \in V$ such that*

1. *All occurrences of x_i are in $\rho(L[v_i])$ for all $1 \leq i \leq k$.*
2. *$x_i \notin \rho(L[v_j])$ for all $i > j$.*
3. *$v_k = q$, where q denotes the root of the directed binary tree G.*

We use the notation $\mathcal{I}(v)$ to denote the maximum value of k such that there exists an independent sequence of message indices in $G[v]$ of length k. In particular, $\mathcal{I}(q)$ denotes the maximum length of an independent sequence of message indices in the tree G. We say that v_i is the *corresponding vertex* of x_i.

Because of condition 2 in Definition 4, the order of independent elements is important. So $(x_2, x_1, x_3, \cdots, x_k)$ might not be an independent sequence, even if $(x_1, x_2, \cdots, x_k)$ is an independent sequence. The definition of the independent sequence coincides with independent elements in a subsequence (see Definition 1) if our tree has a suitable "sequential" structure.

We illustrate Definition 4 with a small example. In Figure 4, $(1, 5, 4)$ is an independent sequence. Here the corresponding vertices of $1, 5$ and 4 are v_1, v_2 and v_3, respectively. However, note that $(4, 1, 5)$ is not an independent sequence since the only vertex v such that all occurrences of 4 are in $\rho(L[v])$ is v_3. One can also check that $(5, 4)$ is still an independent sequence in $G - G[v_1]$ and 1 does not appear in $G - G[v_1]$.

In general, we have the following lemma.

Lemma 6 *If $(x_1, x_2, \ldots, x_k)$ is an independent sequence in G, then $(x_2, \cdots, x_k)$ is also an independent sequence in $G - G[v_1]$, where v_1 is the corresponding vertex of x_1. Also, we have that $x_1 \notin \rho(L[G - G[v_1]])$.*

Proof. $x_1 \notin \rho(L[G - G[v_1]])$ since all occurrences of x_1 are in $\rho(L[v_1])$ (by condition 1 of Definition 4). Also, it is easy to check that $(x_2, \cdots, x_k)$ is an independent sequence in $G - G[v_1]$. ■

Now we can state one of our main theorems of this section. It says that, given a pair (G, ρ) with r independent elements in G, there is a 2^r-way collision attack on the hash function H based on (G, ρ). The complexity of this attack is $O((s+1)2^{n/2})$, where s is the number of intermediate nodes in G. The idea of the attack is very similar to that of Joux's attack. First we try to find r intermediate collison pairs $(m^1_{x_1}, m^2_{x_1}), \cdots (m^1_{x_r}, m^2_{x_r})$. Then we can combine all these pairs independently to obtain a 2^r-way collision attack.

We demonstrate the attack with the example shown in Figure 4.

1. First, fix the message blocks m_2, m_3 and m_6 to be an n-bit string, say IV.

2. Find $m^1_1 \neq m^2_1$ such that $H(v_1, m^1_1) = H(v_1, m^2_1) = h^*_1$ by using $3 \times 2^{n/2}$ computations of f (note that three computations of f are required to obtain a value assigned to v_1).

3. Consider the graph $G_2 = G - G[v_1]$. Find $m^1_5 \neq m^2_5$ such that $H(v_2, m^1_5) = H(v_2, m^2_5) = h^*_2$ by using $3 \times 2^{n/2}$ computations of f (note that three computations of f are required to obtain a value assigned to v_2).

4. Consider the graph $G_3 = G_2 - G[v_3]$ and the mapping $\rho_3(v_1) = h^*_1$, $\rho_3(v_2) = h^*_2$. For this pair (G_3, ρ_3), we can find $m^1_4 \neq m^2_4$ such that $H(v_3, m^1_4) = H(v_3, m^2_4) = h^*_3$ by using $5 \times 2^{n/2}$ computations of f (note that five computations of f are required to obtain a value assigned to v_3, given specified values for h^*_1 and h^*_2).

5. Now it is easy to check that the set

$$\{m : m_i = IV \text{ for } i = 2, 3, 6, m_j = m^1_j, m^2_j \text{ for } j = 1, 4, 5\}$$

   is a multicollision set with the collision value h^*_3. In this example we need $O(11 \times 2^{n/2})$ computations of f. Note that 10 is the number of intermediate nodes.

Now we state and prove a general theorem in detail.

Theorem 7 *There is a 2^r-way multicollision attack on H having complexity $O((s+1)2^{n/2})$, where s is the number of the intermediate nodes in the directed binary tree G and $\mathcal{I}(q) = r$ for the root of the binary tree, q.*

Proof. Suppose that $(x_1, \cdots, x_r)$ is an independent sequence in G. We will find a 2^r multicollision set where each m_{x_i} $(1 \leq i \leq r)$ takes on one of two possible values, and the other m_j's are fixed to be some value IV.

We prove the result by induction on r. Let v_i be the corresponding vertex of x_i. For $r = 1$ this is just a standard birthday attack on H, varying the message block m_{x_1} and fixing all other message blocks by a string IV.

For $r > 1$, we first define the message blocks m_i to be IV for all $i \notin \{x_1, \cdots, x_r\}$. Then we find a pair $(m^1_{x_1}, m^2_{x_1})$ with $m^1_{x_1} \neq m^2_{x_1}$ such that $H(v_1, m^1_{x_1}) = H(v_1, m^2_{x_1}) = h^*_1$ (say). This computation has complexity $t\,2^{n/2}$, where $t = |V[v_1] - L[v_1]|$.

Now consider the graph $G' = G - G[v_1]$ and the MIV $\rho' : L[G'] \to [1, l] \cup \{0,1\}^n$ defined as $\rho'(v_1) = h^*_1$ and $\rho'(v) = \rho(v)$ for any other leaf v in $L[G']$. By Lemma 6, we know that $(x_2, \cdots, x_r)$ is an independent sequence for the hash function based on (G', ρ'). So, by induction, we can find a 2^{r-1}-way collision set

$$\{m : m_j = m^1_{x_i} \text{ or } m^2_{x_i} \text{ if } j = x_i, 2 \leq i \leq r, \text{ otherwise } m_j = IV, \text{ where } j \neq x_1\},$$

with the collision value h^* (say). This computation has complexity $O(|V' - L[G']|)$.

Note that there is no occurrence of index x_1 in the multi-set $\rho'(L[G'])$, and if the intermediate hash value at the vertex v_1 is h_1^*, then the final hash value for (G', ρ') is the same as the final hash value for (G, ρ). Hence,

$$\{m : m_j = m_{x_i}^1 \text{ or } m_{x_i}^2 \text{ if } j = x_i, 1 \le i \le r, \text{ otherwise } m_j = IV\}$$

is a 2^r-way collision set with collision value h^*. The complexity of the attack is

$$O((|V' - L[G']| + |V[v_1] - L[v_1]|)2^{n/2}) = O(|V - L[V]|2^{n/2}) = O((s+1)2^{n/2}).$$

■

Now we prove a simple fact on directed binary trees relevant to out multicollision attack on generalized tree based hash functions. Recall that $S(v)$ denotes the number of indices which appear in $\rho(L[v])$.

Lemma 8 *For any pair (G, ρ) with $S(q) \ge 2N$, there exists a vertex $v \in V$ with $N \le S(v) \le 2N$, where q is the root of the tree $G = (V, E)$.*

Proof. Let $u_1 \to v$ and $u_2 \to v$. Then it is easy to check that $S(v) \le S(u_1) + S(u_2)$. So, if $u_1 \to q$, $u_2 \to q$, then $S(u_1) + S(u_2) \ge 2N$. There will be one vertex, say u_1, with $S(u_1) \ge N$. If $S(u_1) \le 2N$, then the result follows for $v = u_1$. If not, we can continue until we reach a vertex v with $N \le S(v) \le 2N$. ■

Proposition 9 *Let $l = |S(q)|$ where q is the root of the tree. If* $\mathrm{freq}(G) \le 2$, *then there is a vertex v such that $\mathcal{I}(v) \ge N$ or $\mathcal{I}(q) \ge M$ whenever $l \ge 2MN$.*

Proof. We prove the stated result by induction on l. For $M = 1$, the proof is trivial since $\mathcal{I}(q) \ge 1$. So assume $M > 1$. Since $S(q) \ge 2MN \ge 2N$, it follows from Lemma 8 that there is a vertex v such that $N \le S(v) \le 2N$. Now, if $S^1(v) = S(v) \ge N$, then $\mathcal{I}(v) \ge S^1(v) \ge N$. If $S^1(v) < S(v)$, then there is an element, say x_1, which appears exactly twice in $\rho(L[v])$ (note that $\mathrm{freq}(G) \le 2$). Let $G' = G - G[v]$. After we choose an index x_1 in $\rho(L[v])$, we want to make sure that no x_i $(i > 1)$ that is chosen later on also occurs in $\rho(L[v])$. To prevent this from happening, we take all indices of message blocks in $\rho(L[v])$ and "remove" them from any other leaves in the graph, by fixing their values, before applying the inductive hypothesis. Formally, we define $\rho'(u)$ to be an n-bit string, where $u \in \rho(L[v]) \cap \rho(L[G'])$; otherwise, $\rho'(u) = \rho(u)$. Note that $S(G') \ge 2MN - 2N = 2(M-1)N$.

By the induction hypothesis on the graph G', either $\mathcal{I}(q) \ge M - 1$ or there exists a vertex u such that $\mathcal{I}(u) \ge N$. In the latter case, $\mathcal{I}(u) \ge N$ (for the graph G). Otherwise there exist $M - 1$ independent elements, $x_2, \ldots, x_M$, in the graph G'. Also, x_1 does not appear in $\rho(L[G'])$ and $x_2, \cdots, x_M$ do not appear in $\rho(L[v])$. So, $x_1, x_2, \ldots, x_M$ are independent elements in G. ■

Whenever $l \ge 2r^2 n$ either $\mathcal{I}(q) \ge r$ or there is a vertex v such that $\mathcal{I}(v) \ge rn = k$ (say). In the former case, we already have a 2^r-way collision attack. In the latter case, we can do the same thing that we did in the sequential case: Let $(x_1, \cdots, x_k)$ be an independent sequence. Find r vertices $v_1, v_2, \cdots, v_r = q$ in G' $(= G - G[v])$ such that the following occurs:

1. $x_{in+1}, x_{in+2}, \cdots, x_{in+n/2} \in \rho(L(G'[v_i]))$ for all i.
2. $x_{in+1}, x_{in+2}, \cdots, x_{in+n/2} \notin \rho(L(G'[v_j]))$ for all $j < i$.

First, we find a 2^k-way collision on v. Then, we find r successive collisions from the multicollision set. The idea of the attack is very similar with that of the sequential case, so we ignore the details. Our main theorem is as follows.

Theorem 10 *If* $\text{freq}(G(H)) \leq 2$ *then we have a* 2^r*-way multicollision attack having complexity* $\text{O}(r^2\, n\, 2^{n/2})$.

5.1 A Note on Multi-Preimage Attacks

For the sake of completeness, we briefly study the corresponding multi-preimage attacks on generalized sequential or generalized tree-based hash functions. For a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$, we define the *$r$-way preimage (multi-preimage) attack* as follows: Given a random $y \in \{0,1\}^n$, find a subset $\mathcal{C} = \{x_1, \cdots, x_r\}$ of size r (≥ 1) such that $H(x_1) = \cdots = H(x_r) = y$. The complexity of an r-way preimage attack for a random oracle is $\Omega(r\, 2^n)$. On the other hand, for a generalized tree based or sequential hash function there is an r-way preimage attack with complexity $\text{O}(2^n)$. The attack is almost same as the multicollision attack. It starts out exactly the same as the multicollision attack. At the final step, instead of finding a collision, we instead look for outputs having a given value y. The complexity for last step is $\text{O}(2^n)$ which will dominate the $O(r^2\, n\, 2^{n/2})$ complexity of the remaining steps in the attack.

6 Conclusion

Recently there have been many proposed approaches to design hash functions with large output hash values. The most simple and natural one is the concatenation of two classical iterated hash functions. Joux [7] showed some collision (and preimage) attacks on this hash function. It is an interesting question to find design techniques for hash functions for which multicollision attacks are infeasible. In this paper, we have defined a large natural class of hash functions and we studied their security with respect to multicollision attacks. Unfortunately, we have found efficient multicollision attacks on the hash functions when the message blocks are processed at most most twice. So the natural question that arises is if multicollision security an be obtained if the message blocks are used more than twice. One can also search for some other designs outside this class of hash functions, and study their multicollision security.

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

[4] 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).

[5] 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).

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

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

[8] 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`.

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

[10] 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.

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

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

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

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

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

[16] 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`.

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

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

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