Breaking the ICE - Finding Multicollisions in Iterated Concatenated and Expanded (ICE) Hash Functions

Jonathan J. Hoch and Adi Shamir

Department of Computer Science and Applied Mathematics
The Weizmann Institute of Science, Israel

Abstract. The security of hash functions has recently become one of the hottest topics in the design and analysis of cryptographic primitives. Since almost all the hash functions used today (including the MD and SHA families) have an iterated design, it is important to study the general security properties of such functions. At Crypto 2004 Joux showed that in any iterated hash function it is relatively easy to find exponential sized multicollisions, and thus the concatenation of several hash functions does not increase their security. However, in his proof it was essential that each message block is used at most once. In 2005 Nandi and Stinson extended the technique to handle iterated hash functions in which each message block is used at most twice. In this paper we consider the general case and prove that even if we allow each iterated hash function to scan the input multiple times in an arbitrary expanded order, their concatenation is not stronger than a single function. Finally, we extend the result to tree-based hash functions with arbitrary tree structures.

Keywords : hash functions, iterated hash functions, tree based hash functions, multicollisions, cryptanalysis.

1 Introduction

The recent discovery of major flaws in almost all the hash functions proposed so far ([18], [5], [1]) made the analysis of the security properties of these functions extremely important. Some researchers (e.g., Jutla and Patthak [6]) proposed clever ways to strengthen the internal components of standard hash functions in order to make them provably resistant against some types of attacks. A different line of research (which was extensively studied and formalized in Preneel's pioneering work [11]) considered the structural properties of various types of hash functions, assuming that the primitive operations (such as compression functions on fixed length inputs) are perfectly secure. This is similar to the structural study of various modes of operation of encryption schemes, ignoring their internal details.

One of the most surprising results in this area was the recent discovery by Joux [5] of an efficient attack on Iterated Concatenated (IC) hash functions. An iterated hash function has a constant size state, which is mixed with a constant size input by a compression function f to generate the next state. A message of unbounded size is hashed by dividing it into a sequence of message blocks, and providing them one by one to the compression function. The initial state is a fixed IV, and the last state is the output of the hash function. A concatenated hash function starts from several IV's, applies a different compression function to the original message in each chain, and concatenates the final states of all the chains to get a longer output. To prove that multiple chains of compression functions are not much stronger than a single chain, Joux showed how to generate a 2^k−multicollision (i.e., 2^k different messages which are all mapped to the same output value by the hash function) with complexity $k2^{\frac{n}{2}}$. This is only slightly larger than the $2^{\frac{n}{2}}$ complexity of finding one pairwise collision in the underlying compression function via the birthday paradox, and much smaller than the $2^k 2^n$ complexity of finding such a multicollision in a random non-iterated hash function. He then showed how to use multicollisions in F_1 in order to find collisions in the concatenated hash function $F_1(M)\|F_2(M)$ with complexity $O(n2^{\frac{n}{2}})$, which is much smaller than the 2^n complexity of the birthday paradox applied to the $2n-$bit concatenated state. Other possible applications of multicollisions are in the MicroMint micropayment scheme [14] and in distinguishing iterated hash functions from random functions.

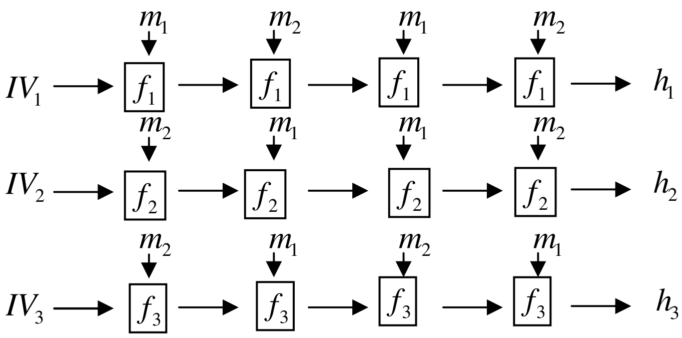

Fig. 1. An example of an ICE hash function, where the output is $h_1\|h_2\|h_3$

One of the simplest ways to overcome Joux's multicollision attack is to use message expansion which forces the iterated hash function to process each message block more than once. For example, the hash function can scan the original message blocks forwards, then backwards, then the even numbered blocks, and finally the odd numbered blocks, before producing the output. In addition, a concatenated hash function can use a different expanded order with each compression function, before concatenating their outputs (see Fig 1). We can assume that the expansion phase increases the total number of message blocks by at most a constant factor s, since higher expansion rates (e.g., quadratic) will make it too expensive to hash long messages, and thus lead to impractical constructions. We call such a generalized scheme an Iterated Concatenated and Expanded (ICE) hash function. Joux's original technique could not handle such functions, since a pair of message blocks which create a collision in a compression function at one point is very unlikely to create another collision later when they are mixed with a different state.

This difficulty was partially resolved in 2005 by Nandi & Stinson [10]. They considered the special case of ICE hash functions in which each message block is used at most twice in the expanded message, and extended Joux's original technique in a highly specialized way to handle this slightly larger class of hash functions.

In this paper we consider the general case of an arbitrary expansion rate s, and show how to find in any ICE hash function whose individual compression functions have $n-$bit states an $O(2^n)$ sized multicollision, using messages whose length is polynomial in n for any constant s. This shows that the Joux multicollision technique is much more powerful and the ICE hash construction is considerably less secure than originally believed.

1.1 Outline of this Paper

The new proof technique is based on careful analysis of the structural properties of sets of words of the form $M' = m_{\alpha_1} m_{\alpha_2} ... m_{\alpha_e}$ which can be derived from the original message $M = m_1 m_2 ... m_l$

by replicating and reordering the message blocks m_i during the expansion phase, when $e \leq sl$. The proof is quite involved, and uses a series of combinatorial lemmas. To make it easier to follow, we first give an overview of the various steps.

The first step is to show that the case of expansion by a total factor s can be reduced to the case of an expansion in which each message block appears at most $q = 2s$ times. The next step of the proof is to reduce such expanded words to the form $\pi_1(M)\|\pi_2(M)...\|\pi_k(M)$ where $k \leq q$ and each π_i is a permutation which contains each message block exactly once. We then show how to construct arbitrarily large multicollisions when the expanded sequence consists of k successive permutations of the message blocks. Finally we show how to use such multicollisions in order to find collisions in the concatenation of several hash functions defined by different sequences.

In section 2 we deal with expansion schemes which can be represented as a sequence of permutations. Section 3 generalizes the proof to any ICE hash function with a constant expansion rate. Section 4 shows how to construct multicollisions when the iterative compression structure is replaced by a tree-like compression scheme. Section 5 summarizes our results and presents some open problems.

2 The Successive Permutations Case

Throughout the paper we denote the set of the first l integers by $L = \{1, 2, ..., l\}$ where $l = |M|$ is the length of the original (unexpanded) message. Where no message is clear from the context, l can be an arbitrary integer. We start by proving a useful lemma:

Lemma 1. *Let B and C be two permuted sequences of the elements of L. Divide B into k consecutive groups of the same size ($\frac{l}{k}$) and name the groups $B_1, ..., B_k$, and divide C into k consecutive groups of the same size ($\frac{l}{k}$) and name the groups $C_1, ..., C_k$. Then for $x > 0$ and $l \geq k^3 x$ there exists a perfect matching of B_i's and C_j's such that $B_i \bigcap C_j \geq x$.*

Proof. We will use the fact that B and C are partitioned into a small number of large disjoint sets, which are likely to have large intersections. We construct the following bipartite graph: $V = \{B_1, ..., B_k, C_1, ..., C_k\}$ and $(B_i, C_j) \in E$ iff $B_i \bigcap C_j \geq x$. According to Hall's matching theorem it is enough to show that any subset of B_i's of size t has at least t neighbors in C, in order to prove that there exists a perfect matching between B and C. Without loss of generality, let $A = B_1 \bigcup ... \bigcup B_t$ be all the elements from a subset of B_i's. Assume for the sake of contradiction that this subset has at most $t-1$ neighbors in C. This means that at most $t-1$ C_j's intersect these B_i's with an intersection of x or more. The maximal number of elements from A which are 'covered' by these elements is $(t-1)\frac{l}{k}$. In addition there are $k-t+1$ C_i's which intersect each of the B_i's in A by less than x. Since there are t B_i's in A, the maximal number of elements in A covered by the remaining C_i's is less than $(k-t+1)tx$. So the total number of elements in A covered by any element from C is less than $(t-1)\frac{l}{k} + (k-t+1)tx$. However, the total number of elements in A is $t\frac{l}{k}$. Taking $l \geq k^3 x$ we have $t\frac{l}{k} \geq txk^2 \geq (t-1)xk^2 + (k-t+1)tx$ for any t. Thus we have a contradiction (not all the elements of A are 'covered') and we conclude that any subset of t B_i's must have at least t neighbors among the C_j's. Hence the conditions from Hall's theorem are fulfilled and there exists a perfect matching between the B_i's and the C_j's. □

Definition 1. *An interval $I = [i_1, i_2]$ is a continuous set of indices $1 \leq i_1 \leq i_2 \leq l$. Then for any sequence α of elements from L, $\alpha[I]$ denotes the subsequence of α defined by $(\alpha_{i_1}, \alpha_{i_1+1}, ..., \alpha_{i_2})$.*

Definition 2. *Let α be some sequence over L and let $X \subseteq L$*
$\alpha|_X$ is constructed as follows: First we take β to be the subsequence of α containing only elements from X. Then we set all consecutive appearances of the same value to a single appearance. For

example, if $\alpha = 1, 2, 3, 3, 2, 4, 2, 3$ *and* $X = \{2, 3\}$ *then we first set* $\beta = 2, 3, 3, 2, 2, 3$ *and then set* $\alpha|_X = 2, 3, 2, 3$.

We now state another useful lemma

Lemma 2. *Let* α *be a sequence over* L *and let* X *be a subset of elements of* L. *If we can construct a* 2^k *Joux multicollision against the hash function based on* $\alpha|_X$ *then we can construct a* 2^k *Joux multicollision against the hash function based on* α.

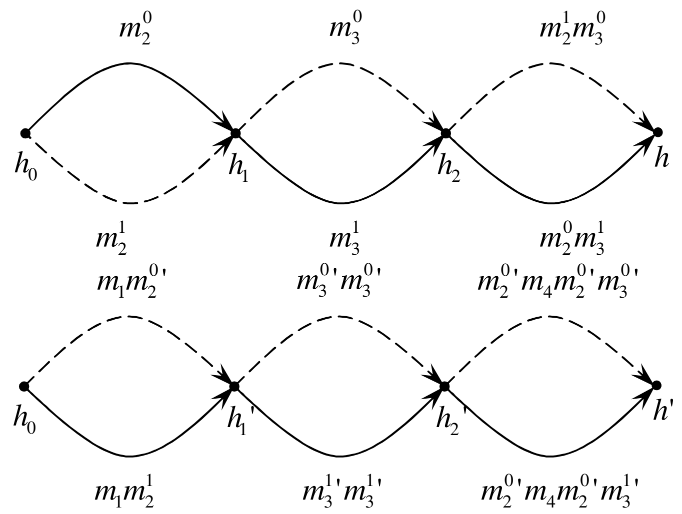

Fig. 2. The Joux multicollision in $\alpha|_X = 2, 3, 2, 3$ (top part) and in $\alpha = 1, 2, 3, 3, 2, 4, 2, 3$ (bottom part, where $X = \{2, 3\}$). Notice how the message blocks are different in $\alpha|_X$ and in α and that all message blocks not in X are set to a constant value. The dotted and solid lines describe the two collision paths in the final 2-collision when $n = 2$

Proof. Let h_0 be the initial hash value. In a Joux multicollision, starting from the initial hash we have a series of intermediate hash values $(h_1, h_2, ..., h_k)$ such that h_i is reachable from h_{i-1} by two different choices for the relevant message blocks. Now let $J_1, J_2, .., J_k$ be the indices of the intervals of message blocks used for the Joux multicollision such that $F(h_{i-1}, M(J_i)) = h_i$ where $M(J_i)$ is the sequence of message blocks corresponding to the indices in the interval J_i. The interval J_i in $\alpha|_X$ corresponds to an interval I_i in the original sequence α such that $\alpha[I_i]|_X = \alpha|_X[J_i]$. Now starting from J_1, we have that there are at least $2^{\frac{n}{2}}$ different messages that can be constructed by changing the message blocks indexed by the indices in J_1, since I_1 includes all of those indices, we can set all other message blocks to a fixed constant and varying only the message blocks indexed by J_1, construct a collision in $F(h_0, I_1^i) = h_1'$ with I_1^0 and I_1^1. The same goes for J_2 and I_2 and so on until J_k and I_k. The important thing to notice is that even when the possible combinations that are used in J_i are not all the combinations, i.e. there are some restrictions stemming from previous use of the message blocks, we still have at least $2^{\frac{n}{2}}$ possible combinations in J_i (which is sufficient for finding a collision with high probability among the different intermediate hash values) and therefore also in I_i. At the culmination of this process we have constructed a 2^k Joux multicollision in the hash function based on α. $\square$

To ease the understanding of the general case of successive permutations, we first give a proof for the special case of 3 successive permutations $\alpha = \pi_1(L)\|\pi_2(L)\|\pi_3(L)$ which is the simplest case which is not treated in [10]. We start by taking a message M of length $\frac{k^3 n^2}{4}$. We now look at the message blocks $\pi_2(L)$ and group them into consecutive groups of size $k\frac{n^2}{4}$. We call the first group B_1 and the last group B_k where 2^k is the size of the multi-collision we are constructing. Similarly we group the message blocks $\pi_3(L)$ into consecutive groups of the same size and name the groups $C_1, ..., C_k$. We use lemma 1 in order to pair each B_i with a unique C_j such that $B_i \bigcap C_j \geq \frac{n^2}{4}$. We now choose from each pair $\frac{n^2}{4}$ message block indices from the intersection and call the union of all the intersections *active indices*, the rest of the message block indices will be called *inactive indices*. Note that since π_2 and π_3 are permutations, each active index occurs in a single pair of B_i and C_j. Let X be the set of all the active indices. According to lemma 2 it suffices to show that we can construct a 2^k Joux multicollision in $\beta = \alpha|_X$. We construct a Joux multicollision on the message blocks indexed by the first part of β (which is taken from $\pi_1(L)$), starting from the initial IV. We then construct a multicollision on the message blocks indexed by the section of β which is taken from $\pi_2(L)$ using intervals containing $\frac{n}{2}$ message blocks each. Finally we construct a multicollision in the message blocks indexed by the section of β which is taken from $\pi_3(L)$ by using intervals containing $\frac{n^2}{4}$ message blocks (which correspond to the C_i's). Notice that the final stage of the construction works because the elements in a specific C_i are all contained in the same interval B_j (and in no other B_t) and thus do not affect the intermediate hash values outside this interval. While the basic idea of using larger and larger blocks in not new (for example, it was used by Joux [5] to compute preimages in generic hash functions), our results generalize the technique and show its real power.

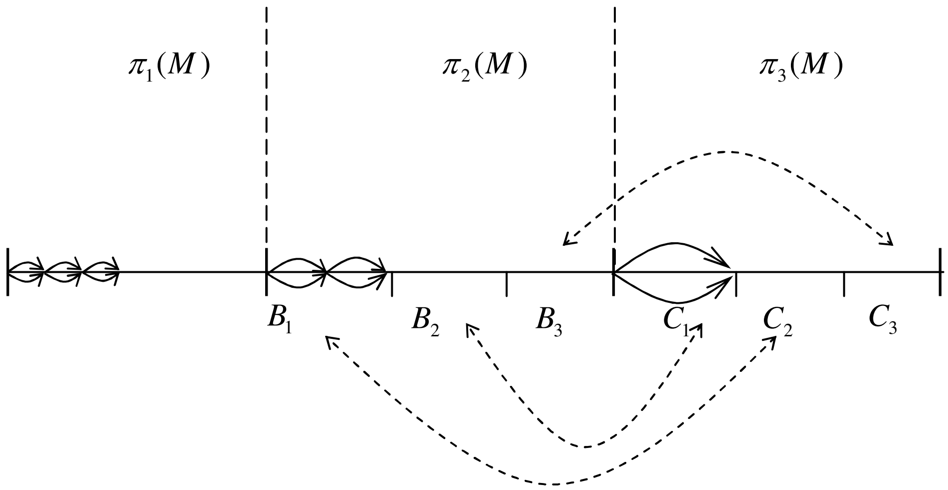

Fig. 3. Multicollision in 3 successive permutations. The dotted lines represent the matching between the B_i's and the C_j's. The solid lines show the collisions built along the way. The collisions in the leftmost section are collisions over single message blocks. The collisions in the middle section are over intervals containing $\frac{n}{2}$ message blocks. The collisions in the rightmost section are over intervals containing $\frac{n^2}{4}$ message blocks

We now prove the general case of successive permutations, by using messages whose length is polynomial in n for any constant expansion rate s.

Theorem 1. *Let α be a sequence of the form $\pi_1(L)\|\pi_2(L)...\|\pi_q(L)$. We can construct a 2^k Joux multicollision against the hash function based on α whenever $l = |M| \geq k^3 n^{3(q-3)+2}$.*

Proof. We start by dividing the last two permutation copies, $\pi_{q-1}(L)$ and $\pi_q(L)$, into k equal length intervals each. We then find a perfect matching between the two sets of intervals as in the 3 permutations case. However, this time we seek an intersection of size $n^{3(q-3)+2}$. After we have our new set of active indices X (which is the disjoint union of the indices from all the intersections), we turn to look at $\alpha|_X$. In this new sequence we examine the permutations $\pi_{q-2}(L)$ and $\pi_{q-1}(L)$. We divide them into kn intervals of equal length and use our lemma to find a perfect matching with an intersection size of $n^{3(q-4)+2}$. We then divide the permutations $\pi_{q-3}(L)$ and $\pi_{q-2}(L)$ into kn^2 intervals and find a perfect matching with an intersection size of $n^{3(q-5)+2}$. We continue downsizing our list of active indices in the same manner until we have found a perfect matching with an intersection size of n^2 between $\pi_3(L)$ and $\pi_2(L)$. The size of X, the set of active indices, starts with $|X| = k^3 n^{3(q-3)+2}$. After the first step we have $kn^{3(q-3)+2}$ remaining active indices, after the second step we have kn segments, each with $n^{3(q-4)+2}$, and after $q-2$ steps we have an intersection size of n^2 for each of the kn^{q-3} segments.

The next stage is to build a Joux 2^k multicollision in the hash function based on $\beta = \alpha|_X$ where X is the final (smallest) set of indices. As in the three permutations case, we start by constructing a Joux multicollision on the $\pi_1(L)$ part of the sequence. We then use intervals of n message blocks to construct a multicollision in the $\pi_2(L)$ part and in general use intervals of size n^{i-1} in the i-th permutation. Since we have k blocks of size n^{q-1} in the last permutation, the process terminates with a 2^k multicollision in the hash function based on β. Using lemma 2, we get a 2^k multicollision in the hash function based on α as required. $\square$

3 Solving the General Case

We now show how to reduce the general case to the successive permutations case. First we state some definitions and prove a useful lemma.

Definition 3. *Let α be a sequence over L:*

$$freq(x, \alpha) = |\{i : \alpha_i = x\}| \tag{1}$$

$$freq(\alpha) = max\{freq(x, \alpha) : x \in L\} \tag{2}$$

Definition 4. *Let $T = t_1, ..., t_t$ be a (not necessary contiguous) sequence of indices in α. Then :*

$$\alpha[T] = \alpha_{t_1}, ..., \alpha_{t_t} \tag{3}$$

In particular if $T = [t_1, t_2]$ is an interval then the definition coincides with definition 1.

Definition 5. *Given any subsequence $\alpha[T]$ of α, we define*

$$S(\alpha[T]) = |\{x \in L : freq(x, \alpha[T]) \geq 1\}| \tag{4}$$

Definition 6. *A set of disjoint intervals $I_1, ..., I_j$ is called* independent *over α if there exists a set of distinct elements $x_1, ..., x_j$ in α such that all the appearances of x_i in α are in $\alpha[I_i]$.*
We will call a set $x_1, ..., x_j$ of distinct elements in α independent *if there exist* independent *intervals $I_1, ..., I_j$ such that all appearances of x_i are in $\alpha[I_i]$.*

Definition 7. *$Ind(\alpha)$ is the largest j such that there exists a set $I_1, ..., I_j$ which is independent over α.*

For example $\alpha = 1, 2, 1, 3, 2, 4, 2, 4$ has $Ind(\alpha) = 3$ by taking the independent elements $1, 3, 4$. We can see for example that the smallest interval containing all the appearances of 4 does not contain either 1 or 3. However, we cannot chose $1, 2, 3, 4$ as independent elements since they are interleaved in α.

Definition 8. *A left-end interval is an interval of the form $I = [1, i]$ for some integer i.*

In Nandi and Stinson's paper[10] the authors proved and used the following lemma (translated into our notation):

Lemma 3. *Let α be a sequence of elements from L with $freq(\alpha) \leq 2$ and $S(\alpha) = l$. Suppose that $l \geq MN$. Then at least one of the following holds:*

1. *$Ind(\alpha) \geq M$, or*
2. *there exists a left-end interval I such that $Ind(\alpha[I]) \geq N$.*

The generalization we wish to prove in order to handle arbitrary ICE hash functions is as follows:

Lemma 4. *Let α be a sequence of elements from L with $freq(\alpha) \leq q$ and $S(\alpha) = l$. Suppose that $l \geq MN$. Then at least one of the following holds:*

1. *$Ind(\alpha) \geq M$, or*
2. *there exists a left-end interval I and a subset $X \subseteq L$ s.t. $freq(\beta) \leq q - 1$ and $S(\beta) \geq \frac{N}{q-1}$ where $\beta = \alpha[I]|_X$.*

Proof. The proof follows the same general lines as in [10], and uses induction on l. For the left-end interval $I = [1, N]$ either $freq(\alpha[I]) \leq q - 1$ or there exists an element x_1, which appears q times in the sequence $\alpha[I]$. If the former holds then we have N elements in $\alpha[I]$ and each one of them can occur at most $q - 1$ times, and thus the number of distinct elements $S(\alpha[I])$ is at least $\frac{N}{q-1}$. We set $X = L$ and $\beta = \alpha[I]|_X = \alpha[I]$ and we are done. So we assume that there exists an element x_1, which appears q times in the sequence $\alpha[I]$. We remove all elements from α which appear in $\alpha[I]$ and call the new sequence $\alpha_1 = \alpha[I_1]$ for some set of indices I_1.
Note that $S(\alpha_1) \geq MN - N = (M - 1)N$ since we have removed at most N distinct elements from α. By the induction hypothesis, either $Ind(\alpha_1) \geq M - 1$ or there exists a left-end interval J and a subset X of L such that $freq(\beta[J]) \leq q - 1$ and $S(\beta[J]) \geq \frac{N}{q-1}$ where $\beta = \alpha[J]|_X$. In the latter case we simply take X and β as provided from the lemma and set the interval I to be the shortest left-end interval containing J. In the former case let $I_2, ..., I_M$ be an independent set of intervals over α_1 containing the independent indices $x_2, .., x_M$. These intervals can be mapped to independent intervals $J_2, ..., J_M$ over α where J_i is the minimal interval containing all the occurrences of x_i for $i = 2, .., M$. Notice that $x_1 \notin J_i$ for $i = 2..M$ since all appearances of x_1 are before the first index of α_1 so we can add an interval $J_1 = 1..N$ to the list of independent intervals and now we have that $Ind(\alpha) \geq M$ as required. □

Now we prove one final lemma before turning to prove our main theorem. We want to prove by induction on q the following claim:

Lemma 5. *For any integer x, given a sequence α with $freq(\alpha) \leq q$ and $S(\alpha)$ large enough, we can find a subset of indices X, $|X| \geq x$ such that $\alpha|_X$ is in the form of up to q successive permutations over the same set of indices X.*

Proof. Let $f_q(x)$ be the minimal alphabet size of a sequence α with $freq(\alpha) \leq q$ that ensures that there is a subset of indices X, $|X| \geq x$ such that $\alpha|_X$ is in the form of successive permutations. We will prove that $f_q(x) \leq C_q x^{D_q}$, for some constants C_q, D_q which increase with q.

We start by claiming that $f_1(x) = x$ (i.e., $C_1 = D_1 = 1$), since any sequence α with $S(\alpha) = x$ and $freq(\alpha) = 1$ is a single permutation of all the indices that occur in α. For notational purposes we will define $f_0(x) = 0$ for all x. Now assume that we have proven the inequality $f_k(x) \leq C_k x^{D_k}$ for all $k < q$. Given a sequence α such that $S(\alpha) \geq x(q-1)f_{q-1}(f_1(x) + f_2(x) + ... + f_{q-1}(x))$, we apply lemma 4 with $M = x$ and $N = (q-1)f_{q-1}(f_1(x) + f_2(x) + ... + f_{q-1}(x))$. There are now two cases. In the first we have $Ind(\alpha) \geq x$, and let X be the set of all independent indices. By definition we have $|X| = Ind(\alpha) \geq x$ and $\alpha|_X$ is a single permutation of the indices in X (since $freq(\alpha|_X) = 1$). In the second case we have a left-end interval I and a subset X' such that $freq(\alpha[I]|_{X'}) \leq q-1$ and $S(\alpha[I]|_{X'}) \geq \frac{N}{q-1} = f_{q-1}(f_1(x) + ... + f_{q-1}(x))$. Now using the inductive hypothesis on $\alpha[I]|_{X'}$ we get a subset X'' such that $|X''| \geq f_1(x) + ... + f_{q-1}(x)$ and $\alpha[I]|_{X''}$ is in successive permutations form with at most $q-1$ permutations. Using the pigeonhole principle we see that there must exist an $0 \leq i \leq q-1$ such that at least $f_i(x)$ indices appear exactly i times in the remainder of $\alpha|_{X''}$. We set X''' to be that subset of indices and apply our induction hypothesis on the remainder of $\alpha|_{X'''}$ (after the interval I). We remain with a subset X, $|X| \geq x$ such that $\alpha|_X$ is in successive permutations form with at most i permutations. Now notice that each index appeared at most q times in α so the number of permutations is at most q. We have shown that

$$
\begin{aligned}
f_q(x) &\leq x(q-1)f_{q-1}(f_1(x) + f_2(x) + ... + f_{q-1}(x)) && (5)\\
&\leq x(q-1)f_{q-1}((q-1)f_{q-1}(x)) && (6)\\
&\leq x(q-1)f_{q-1}((q-1)C_{q-1}x^{Dq-1}) && (7)\\
&\leq x(q-1)C_{q-1}(q-1)^{D_{q-1}}C_{q-1}^{D_{q-1}}x^{D_{q-1}^2} = C_q x^{D_q} && (8)
\end{aligned}
$$

for $C_q = (q-1)^{D_{q-1}+1}C_{q-1}^{D_{q-1}+1}$ and $D_q = D_{q-1}^2 + 1$. This proves the induction hypothesis for q. □

Finally we put all the building blocks together to prove the theorem:

Theorem 2. *Let α be any sequence over L with $|\alpha| \leq sl$ (where $l = |L|$ and s is the constant expansion factor). Then we can compute a 2^k multicollision in the hash function based on α with time complexity $O(poly(n,k)2^{\frac{n}{2}})$.*

Proof. We start with a sequence α over L of length at most sl. There must be a subset of $\frac{l}{2}$ indices, each appearing at most $q = 2s$ times in α. Since otherwise we would have more than $\frac{l}{2}$ indices each appearing at least $2s$ times, giving more than $\frac{l}{2}2s = sl$ elements in the sequence. Let X be the set of these indices. According to lemma 2 it is enough to show that we can construct a Joux multicollision against the hash function based on $\alpha|_X$. Notice that $freq(\alpha|_X) \leq q$. We now apply lemma 5 and we get a subset X', $|X'| \geq k^3 n^{3(q-3)+2}$ such that $\alpha|_{X'}$ is in successive permutations form and $freq(\alpha|_{X'}) \leq q$. According to theorem 1, we can now construct a 2^k multicollision in the hash function based on $\alpha|_{X'}$ and according to lemma 2, we can construct a multicollision in the hash function based on α. □

3.1 Constructing a Collision in an ICE Hash Function

Constructing a collision in a concatenation of two iterated and expanded functions is done by following the recipe presented by Joux. We first construct a $2^{\frac{n}{2}}$ multicollision in the first function and then rely on the birthday paradox to find a collision among the $2^{\frac{n}{2}}$ values of the second hash function on the messages used in the multicollision. However, generalizing the result for 3 or more functions is not as easy.

As you recall the intermediate hash values of an iterated and expanded hash function based on a sequence α are calculated by $h_i = f(h_{i-1}, m_{\alpha_i})$. However, we have not used in our proof the fact that the compression function f is the same in each step. In fact, we do not need this fact and can generalize the calculation of the intermediate hash values to $h_i = f(i, h_{i-1}, m_{\alpha_i})$. We will now show how to construct a collision in an ICE hash function based on three sequences $\alpha_1, \alpha_2, \alpha_3$ and corresponding hash functions F_1, F_2, F_3. The construction we show is easily generalized to an arbitrary number of hash functions.

We will look at the sequence $\alpha = \alpha_1 \| \alpha_2$. The first step is to find a set X such that $\alpha_2|_X$ is in successive permutations form. We then find a subset $X' \subseteq X$ such that $\alpha_1|_{X'}$ is in successive permutations form. Notice that $\alpha_2|_{X'}$ will still be in successive permutations form. We now construct a $2^{\frac{n}{2}}$ Joux multicollision in the sequence $\alpha|_{X'}$ which is also in successive permutations form (as the concatenation of two such sequences). The important point is that the sequence of intervals $I_1, ..., I_k$ which form the multicollision, does not have any interval which spans the border between α_1 and α_2. Taking this sequence of intervals we can now construct a $2^{\frac{n}{2}}$ simultaneous multicollision in the hash functions F_1 and F_2. With such a large multicollision we can find with high probability a pair of messages which hash to the same value also under F_3. Thus we have found a collision in the ICE hash function $F_1(M) \| F_2(M) \| F_3(M)$ with complexity $O(poly(n)2^{\frac{n}{2}})$ instead of the expected $2^{\frac{3n}{2}}$ from the birthday paradox. A simple extension of the idea can handle the concatenation of any constant number of hash functions.

4 Tree Based Hash Functions

We now turn our attention to a more general model for constructing hash functions which we call TCE (Tree based, Concatenated, and Expanded). As in the iterated case we will base our analysis on the model presented in [10]. A tree based hash function uses a binary tree $G = (V, E)$ where the leaves are at the top and the root at the bottom. The leaves are labeled by message block indices or constant values. Given a message M, $F_G(M)$ is computed as follows: the label for each non-leaf x is computed by applying the compression function f to the two nodes directly above x. The label of the root is the output of the hash function. Note that tree based hash functions include iterated hash functions as a special case, by using trees with a single IV to root path, and hanging all the messages blocks off this path. In [10] the authors treated the special case in which every index appears at most twice in the leaves of the tree. We generalize this result to any constant number of appearances.

Definition 9. *Let $v \in V$ be a vertex in G, $W(v)$ is the set of all leaves in the subtree rooted at v.*

Definition 10. *If v is a leaf then $\rho(v)$ is its label (the index of the corresponding message block), and $\rho(v_1, ..., v_k)$ is the sequence $\rho(v_1)...\rho(v_k)$.*

In the following definitions we redefine some of the notations used in the iterated case to apply to trees. When using the definitions we will sometimes abuse notation and use interchangeably a tree G and its root r. For example we write $Ind(v)$ when meaning $Ind(G')$ where G' is the subtree rooted at v.

Definition 11. *Let r be the root of G. An independent vertex sequence is an* ***ordered*** *sequence of vertices $v_1, ..., v_k$ such that there exists a sequence of leaves $w_1, ..., w_k$ satisfying the following conditions:*

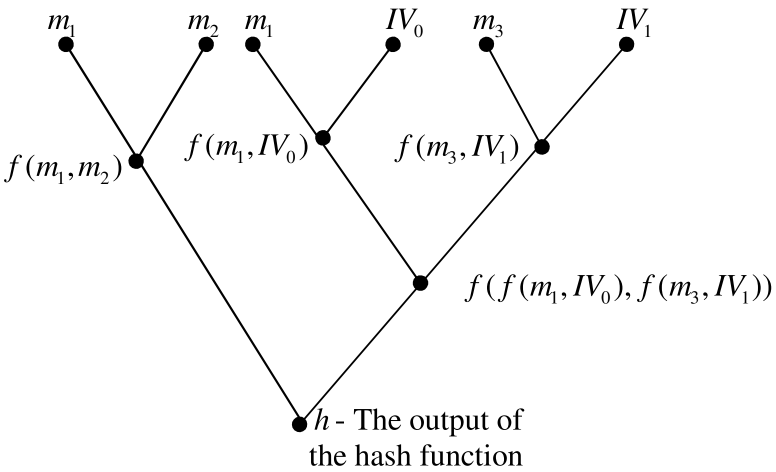

Fig. 4. A example of a TCE

1. *All appearances of* $\rho(w_i)$ *are in* $\rho(W(v_i))$
2. $j < i \Longrightarrow \rho(w_i) \notin \rho(W(v_j))$
3. $v_k = r$

The maximal length of an independent vertex sequence in G *is denoted* $Ind(G)$.

Definition 12. *Let* r *be the root of* G.

1. $S(G)$ *is the number of distinct labels in* $\rho(W(r))$
2. $freq(G) = freq(\rho(W(r)))$ *where* $\rho(W(r))$ *is treated as a sequence.*

Definition 13. *Let* G *be a tree with a whose leaves are labeled by elements from* L *and let* $X \subseteq L$. $G|_X$ *is the pruned tree resulting from the following process:*

1. *Delete from* G *all the original leaves which have labels not in* X.
2. *Repeatedly delete from* G *any newly created leaf which is unlabeled.*

Before we start the technical proof, we will give an overview of what is coming and show the correspondence between the proof of the tree-based case and the proof of the iterated case. As in the previous proof, we first want to reduce the general case to a case equivalent to the successive permutations case.

Definition 14. *A tree* G *is in 'successive permutations' form (with* r *'permutations') if we have a set of vertices* $v_1, ..., v_r$ *s.t.* $S(v_1) = ... = S(v_r) = S(G)$ *and* $Ind(W(v_i) \setminus \bigcup_{j<i} W(v_j)) = S(G)$.

Each vertex v_i corresponds to a permutation in the iterated case, and contains in its leaves all the variables. Furthermore, if we look at the subtree rooted at v_i and remove all smaller subtrees rooted at v_j, then we can construct an independent sequence using all the indices with the root at v_i. The definition is best understood if we think about the iterated case as a special case of a tree with a single path from IV to the root.

The next step is to show that we can construct a multicollision in this special tree structure. The proof will be very similar to the one in the iterated case but with one additional component.

In a tree we have to ensure that when taking a set of indices and trying to get a collision by changing their values, they actually have a common root which is compatible with the other groups of indices. We will later prove a lemma to this effect, and use it to prove the tree version of lemma 4 and then the tree version of lemma 5.
We start by proving the reduction from the general case to the 'successive permutations' case.

Lemma 6. *Given a tree G with $S(G) \leq 2MN$ and $freq(G) \leq q$, at least one of the following claims is true:*

1. *$Ind(G) \leq M$ or*
2. *there exists a node v and a subset X such that $freq(G|_X(v)) \leq q-1$ and $S(G|_X(v)) \leq N$*

Proof. We extend the proof of a similar lemma from [10]. We first note that in any binary tree G such that $S(G) \geq 2N$, there exists a vertex v such that $N \leq S(v) \leq 2N$. We now prove the result by induction on $l = S(G)$. For the basis of the induction we take $M = 1$ and we have that $Ind(G) \geq 1$ is always true. Now assume that we have proved the lemma for all values less than l. Since $S(G) \geq 2MN \geq 2N$ we know from the observation above that there exists a vertex v such that $N \leq S(v) \leq 2N$. Now if $freq(v) \leq q-1$ then we are done, since we set $X = L$ and we have that $S(G|_X(v)) \geq N$ and $freq(G|_X(v)) \leq q-1$. Otherwise we have an element x_1 such that x_1 appears q times in $\rho(W(v))$. We define $G' = G \setminus \{v\}$ (removing the subtree rooted at v) and set X to be $\rho(G')$. $S(G') \geq 2MN - 2N = 2(M-1)N$, so by the induction hypothesis we have that either $Ind(G') \geq M-1$ or there exists v' and X' such that $S(G'|_{X'}(v')) \geq N$ and $freq(G'|_{X'}(v')) \leq q-1$. If the later happens then we simply set $X = X'$ and $v = v'$ and we are done. Otherwise we have an independent sequence $x_2, ..., x_M$ in G' but since x_1 appears only in $W(v)$, we can add x_1 to our list of independent indices and have $Ind(G) \geq M$. □

One of the differences between the iterated case and the tree case is that in a tree it is not sufficient to find a group of message blocks which can be varied independently in order to find a collision. In a tree these blocks must have a common root in which the collision will be formed. In the following lemma we prove that we can always find suitable groups of message blocks in the tree.

Lemma 7. *Given a tree G s.t. $freq(G) = 1$ and $S(G) \geq (2k-1)x$ we can find k distinct nodes, $v_1, ..., v_k$, such that $W(v_i) \nsubseteq W(v_j)$ whenever $i > j$, and $S(W(v_i) \setminus \bigcup_{j<i} W(v_j)) \geq x$.*

Proof. We prove the claim by induction on k. For $k = 1$ we have a tree G with $freq(G) = 1$ and $S(G) \geq x$, and setting v_1 to the root of G satisfies the lemma. Now we assume that we have proved the lemma for all positive integers less than k. Given a tree G with $S(G) \geq (2k-1)x \geq 2x$ we find a node v' with $x \leq S(v') \leq 2x$. Let G' be the subtree rooted at v' and let the tree $G'' = G \setminus G'$ be the result of removing all nodes of G' from G. Notice that $S(G') \geq 2(k-1)x - 2x = (2(k-1)-1)x$. Using the induction hypothesis on G'' we have $k-1$ vertices $v_2, ..., v_k$ such that $W(v_i) \nsubseteq W(v_j)$ whenever $i > j$ and $S(W(v_i) \setminus \bigcup_{j<i} W(v_j)) \geq x$. Now setting $v_1 = v'$ we get a full set $v_1, ..., v_k$ as required since $S(G') \geq x$. □

Theorem 3. *Given a tree based hash function F_G based on the tree G, we can find a 2^k multi-collision whenever G is in 'successive permutations' form and $S(G) \geq 2^{\frac{q(q-1)}{2}} k^3 n^{3(q-3)+2}$.*

Proof. The idea of the proof is the same as in the iterated case, the only difference is that we have to make sure that when choosing a group of message blocks, they indeed have a common root (high enough in the tree) where they can form a collision. The main step in the iterated case was finding a perfect matching between two permutations. In the tree case we also need to make sure that each segment of indices has a common root which doesn't interfere with the other segments of indices. After finding the first matching, we have to find k distinct nodes as in lemma 7 in each 'permutation' copy. We start by finding such a sequence in the first 'permutation' copy, and this reduces the number of active indices by a multiplicative factor of 2. From the remaining indices we need to find a vertex sequence in each of the other 'permutation' copies such that all the variables will be the same. This way we lose a total factor of 2^q for the q 'permutations'. In the second step we have kn segments in the matching. This time however we don't care what happens in the last permutation since we only need the larger structure of k segments. So this time we lose a factor of 2^{q-1} to make sure that all the segments of indices have the required common roots. Continuing for the $q-1$ steps we see that we lose a factor of $2^{\frac{q(q-1)}{2}}$. So the required size of $S(G)$ is the same as in the iterated case except for a factor of $2^{\frac{q(q-1)}{2}}$. Once we have $S(G) \geq 2^{\frac{q(q-1)}{2}} k^3 n^{3(q-3)+2}$ we can carry out the same construction as in the iterated case, where between the steps we use lemma 7 to ensure that the remaining indices have the required structure.

As in the iterated case we need a lemma saying that it is ok to set unselected message blocks to constants.

Lemma 8. *Let G be a tree over L, and let X be a subset of indices. If we can construct a 2^k Joux multicollision against the hash function based on $G|_X$ then we can construct a 2^k Joux multicollision against the hash function based on G.*

The proof of this lemma follows the same lines as in the iterated case. We have one more lemma to prove in order to create all the building blocks needed for the general case.

Lemma 9. *Given a tree G with $freq(G) \leq q$ we can find a subset of indices X such that $G|_X$ is in the form of 'successive permutations'.*

The proof is practically the same as in the sequential case and is omitted here due to space limitations. The only difference is that lemma 6 is used instead of lemma 4. We can now sketch the proof for the general case.

Theorem 4. *Given a tree based hash function F_G based on the tree G. We can find a 2^k multicollision whenever there exists a constant q such that $freq(G) \leq q$ and $S(G) \geq poly_q(n,k)$ in time complexity $O(poly(n,k)2^{\frac{n}{2}})$.*

Proof. We start with a tree G over L with $freq(G) \leq q$. We now apply lemma 9 and we get a subset X, $|X| \geq 2^{\frac{q(q-1)}{2}} k^3 n^{3(q-3)}$ such that $G|_X$ is in 'successive permutations' form and $freq(G|_{X'}) \leq q$. We can now construct a 2^k Joux multicollision in the hash function based on $G|_{X'}$ and according to lemma 8, we can construct a Joux multicollision in the hash function based on G. □

Due to space limitations we omit the full description of finding a collision in a TCE hash function, which is a concatenation of the outputs of several trees. However we can use a procedure analogous to the one used in the iterated case to show that we can find a collision in a general TCE hash function in time $O(poly(n)2^{\frac{n}{2}})$.

5 Summary

We have shown that a large class of natural hash functions (ICE and its generalization TCE) is vulnerable to a multicollision attack, and we hope that the techniques developed here will help in creating multicollision attacks against even more complicated types of hash functions. For example, a different type of message expansion which would be interesting to examine can use linear mixing of the message blocks, instead of pure repetition of the message blocks. Other research directions are to find other countermeasures against the Joux multicollision attack such as the scheme suggested by Lucks [9], or finding additional uses of multicollisions as building blocks in more general attacks as in [5], [7] and [8].

6 Acknowledgments

The authors would like to thank Mridul Nandi and Douglas Stinson whose paper[10] motivated our research and contributed to its development. In addition, we would like to thank the anonymous referees for helping clarify the presentation and pointing out some minor errors.

References

1. E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet & W. Jalby, *Collisions of SHA-0 and Reduced SHA-1*, Eurocrypt 2005
2. J. Daemen, R. Govaerts & J. Vandewalle, *A Framework for the Design of One-Way Hash Functions Including Cryptanalysis of Damgrd's One-Way Function Based on a Cellular Automaton*, Asiacrypt 1991
3. A. De Santis & M. Yung, *On the Design of Provably Secure Cryptographic Hash*, Eurocrypt 1990
4. H. Gilbert & H. Handschuh, *Security Analysis of SHA-256 and Sisters*, Selected Areas in Cryptography 2003 NIST Cryptographic Hash Workshop 2005
5. A. Joux, *Multicollisions in Iterated Hash Functions*, Crypto 2004.
6. C. Jutla & A. Patthak, *A Simple and Provably Good Code for SHA Message Expansion*, IACR preprint archive
7. J. Kelsey & B. Schneier, *Second Preimages on n-bit Hash Functions for Much Less than* 2^n *Work*, Eurocrypt 2005.
8. J. Kelsey & T. Kohno, *Herding Hash Functions and the Nostradamus Attack*, NIST Cryptographic Hash Workshop 2005
9. S. Lucks, *Design Principles for Iterated Hash Functions* IACR preprint archive
10. M. Nandi & D. R. Stinson, *Multicollision Attacks on a Class of Hash Functions* , IACR preprint archive
11. B. Preneel, *Analysis and design of cryptographic hash functions*, PhD thesis, Katholieke Universiteit Leuven (Belgium), 1993.
12. B. Preneel, R. Govaerts & J. Vandewalle, *Hash Functions Based on Block Ciphers: A Synthetic Approach*, Crypto 1993
13. B. Preneel, *Design Principles for Dedicated Hash Functions*, Fast Software Encryption 1993
14. R. Rivest & A. Shamir, *PayWord and MicroMint: Two simple micropayment schemes*, CryptoBytes, volume 2, number 1
15. P. Rogaway & T. Shrimpton, *Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance*, Fast Software Encryption 2004
16. X. Wang, X. Lai, D. Feng, H. Chen & X. Yu, *Cryptanalysis of the Hash Functions MD4 and RIPEMD*, EUROCRYPT 2005
17. X. Wang, H. Yu & Y. Yin, *Efficient Collision Search Attacks on SHA-0* ,Crypto 2005
18. X.Wang, Y. Yin & H. Yu, *Finding Collisions in the Full SHA-1 Collision Search Attacks on SHA1*, Crypto 2005