PROPOSALS FOR ITERATED HASH FUNCTIONS

Lars R. Knudsen
Technical University of Denmark
Department of Mathematics, Building 303, 2800 Kgs. Lyngby, Denmark

Søren S. Thomsen
Technical University of Denmark
Department of Mathematics, Building 303, 2800 Kgs. Lyngby, Denmark

Keywords: Cryptographic hash functions, Merkle-Damgård constructions, multi-collisions, birthday attacks.

Abstract: The past few years have seen an increase in the number of attacks on cryptographic hash functions. These include attacks directed at specific hash functions, and generic attacks on the typical method of constructing hash functions. In this paper we discuss possible methods for protecting against some generic attacks. We also give a concrete proposal for a new hash function construction, given a secure compression function which, unlike in typical existing constructions, is not required to be resistant to all types of collisions. Finally, we show how members of the SHA-family can be turned into constructions of our proposed type.

1 INTRODUCTION

Attacks on hash functions can broadly be divided into two categories; generic attacks on hash function constructions, and so-called short-cut attacks that exploit weaknesses of specific hash functions. The last few years have seen a fairly large number of attacks of both kinds. In this paper, we particularly focus on generic attacks. We propose a new hash function construction which we believe protects against existing generic attacks, and we argue why it will also complicate short-cut attacks on existing hash functions modified to be based on this construction.

A common method of constructing hash functions taking an input of arbitrary size is to iterate a number of times over a so-called compression function $f : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$ taking only fixed-length input. The message d is split into a number of blocks d_i of equal size m. To do this, the message must in general be padded, and this is usually done by always appending a '1'-bit to the message, then a suitable number of '0'-bits, and finally the length of the original message is appended. When the message is split into blocks $d_1 \| d_2 \| \ldots \| d_s$, each block can be processed individually by the compression function f. An initial n-bit value h_0 is defined for the hash function, and then subsequent chaining variables are computed as $h_i = f(h_{i-1}, d_i)$, $i = 1, 2, \ldots, s$. We shall refer to this as the Merkle-Damgård construction (Damgård, 1989; Merkle, 1989).

In this paper, we propose a new method of constructing a hash function from a compression function. The new method has some attractive properties. For instance, it does not require the compression function to be completely collision resistant in order for the hash function to be so. It makes some generic attacks much harder to mount, and it also seems to protect very well against known short-cut attacks such as recent attacks on MD5 (Wang and Yu, 2005) and SHA-1 (Biham et al., 2005; Wang et al., 2005). In general, it leaves an attacker with much less freedom, and the amount of freedom that an attacker has to choose messages in existing hash functions is, we believe, a very important reason for the failure of these hash functions to achieve collision resistance. We discuss possible methods of complicating the generic attacks that our proposal in its bare form does not protect against.

In the following we denote by *the SHA-family of hash functions* the collection of hash functions of (FIPS180b, 1995) and (FIPS180c, 2002), i.e. SHA-1, SHA-256, SHA-512, SHA-224, and SHA-384.

2 PROPERTIES OF EXISTING CONSTRUCTIONS

In this section we consider some of the generic attacks on the Merkle-Damgård construction as well as on the

R. Knudsen L. and S. Thomsen S. (2006).
PROPOSALS FOR ITERATED HASH FUNCTIONS.
In *Proceedings of the International Conference on Security and Cryptography*, pages 246-253
DOI: 10.5220/0002102102460253

underlying Davies-Meyer construction.

2.1 The Davies-Meyer Construction

The compression functions of most popular hash functions, including the SHA-family, in use today are of the form $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$,

$$h_i = f(h_{i-1}, d_i) = p(h_{i-1}, d_i) + h_{i-1}, \quad (1)$$

where p is a bijective mapping on n bits for fixed value of d_i. This construction was known as the Davies-Meyer construction for many years, but was since contributed to Matyas and Meyer (Preneel, 1993). Such functions are traditionally built from iterating a relatively weak function, say g, a number of times, $p(x, y) = g \circ g \circ \cdots \circ g(x, y)$. If the size of the range of g would be less than 2^n, then the size of p would decrease with the number of g invocations. Therefore, g is a bijection in most designs. An invertible compression function is however not always desirable which is why one adds one of the two inputs of p to the output. This design is very similar to the designs of modern block ciphers, which are typically constructed from iterating a weak function several times. Indeed the compression functions from the SHA-family can be used for encryption (Handschuh et al., 2001).

It is well-known (Preneel, 1993) though that the Davies-Meyer construction has an unfortunate property, which adds to the success of the 2nd preimage attack of Section 2.2.3.

- Let $p_y(h_{i-1})$ denote $p(h_{i-1}, d_i)$ for a fixed value of $d_i = y$.
- Choose any m-bit value y and any n-bit value α.
- Compute $h_{i-1} = p_y^{-1}(\alpha)$.

It follows that $p(h_{i-1}, y) = \alpha$. Consequently one gets $h_i = f(h_{i-1}, d_i) = h_{i-1} + \alpha$. E.g., by choosing $\alpha = 0$ one gets $h_i = h_{i-1}$ which is called a *fixed point* for the compression function f.

2.2 The Merkle-Damgård Construction

Collision resistance of a compression function is extended to the hash function when using the Merkle-Damgård construction, as proved by the well-known Theorem 1.

Theorem 1. *Let H be a hash function based on the Merkle-Damgård construction with length padding (also known as MD-strengthening) and compression function f. Then a collision of H implies a collision of f.*

A number of attacks on the general Merkle-Damgård construction show that if the compression function fails, then the entire hash function goes down. Some of these attacks are now described.

2.2.1 Multi-collisions

In (Joux, 2004), Antoine Joux described a new method for constructing multi-collisions from a number of single collisions. A multi-collision is a set of messages, all having the same hash value. If an n-bit hash function produces random hash values, then the complexity of a multi-collision attack consisting of r messages is about $2^{(r-1)n/r}$. Joux shows that in the Merkle-Damgård construction, this complexity can be reduced to $(\log_2 r)2^{n/2}$.

The method is quite simple. It requires access to a machine $\mathcal{C}$ that finds ordinary one-block collisions of a hash function H given any initial value. Let the initial value of H be h_0, and let f be the compression function of H. f accepts two inputs, a chaining variable and a message block. Obtain a collision (d_1, d_1') from $\mathcal{C}$ with h_0 as the initial value, i.e. $f(h_0, d_1) = f(h_0, d_1')$. Call this value h_1, i.e. $h_1 = f(h_0, d_1)$. Now, obtain a second collision (d_2, d_2') from $\mathcal{C}$ with h_1 as initial value. Repeat this $t = \log_2 r$ times. We now have t pairs of messages, where for each pair we can select an arbitrary member of the pair to construct a message of t blocks, and all such messages have the same hash value. There are $r = 2^t$ ways to select such a message, an hence we have an r-way multi-collision. The complexity of the attack is t times the complexity of finding a single collision with $\mathcal{C}$, and this is at most $2^{n/2}$ for any hash function.

2.2.2 The Herding Attack

The herding attack (Kelsey and Kohno, 2006) by Kelsey and Kohno bears some resemblance to the multi-collision attack of Joux. From a large (power of two) number $N = 2^\ell$ of chaining variables, colliding pairs of messages forming a binary collision tree are found such that by each level in the tree, the number of chaining variables is reduced by a factor two. At the root of the tree is the chaining variable h. From the N chaining variables there are N messages of length ℓ blocks, all having the hash value h. This tree of collisions can be used as follows.

Say an adversary would like to falsely claim that he is in possession of some knowledge. He constructs the collision tree described above, and publishes h. Some time later, when he acquires the knowledge, he forms a message d_I containing the information, and computes its intermediate hash. He then searches for a linking message d_L, such that the intermediate hash of $d_I \| d_L$ is among the N chaining variables at the top of the collision tree. From this chaining variable, he selects the set of message blocks that "herds" the entire message to the hash value h.

Note that the kind of collisions needed for this attack is different from the multi-collisions found using the method of Joux. In this case, the multi-collision must form a binary tree. Such multi-collisions seem harder to find than those of Joux, however, a method faster than brute force is given in (Kelsey and Kohno, 2006).

2.2.3 2nd Preimage Attack

In (Kelsey and Schneier, 2005) a second preimage attack on the general Merkle-Damgård construction was presented. This attack makes use of what the authors call expandable messages – a set of messages of *different* lengths, which all have the same intermediate hash given a certain initial value and not including MD-strengthening. In the Merkle-Damgård construction, any message of the expandable message set can be replaced with any other message from the set, without changing the intermediate hash value. To be more precise, given a hash function H with compression function f and initial value h_0, assume that $(\mu_1, \mu_2, \ldots, \mu_k)$ are k messages of lengths 1 to k, all producing the same intermediate hash value given the initial value h_0. Then $H(\mu_i \| x) = H(\mu_j \| x)$ for any x and any $0 < i, j \leq k$ *when length padding is omitted*.

Consider again the hash function H, and let its length be n. If one has produced an expandable set $\mathcal{E}$ of messages with the initial value h_0, then a second preimage attack can be performed as follows. Given a very long message M (whose second preimage we are looking for), compute a list L of all (except the first few) of the intermediate hashes of M, i.e. all the chaining variables that are produced when M is hashed. If $f(h_0, d_i) = h_e$ for all $d_i \in \mathcal{E}$, then look for a message d_L for which $f(h_e, d_L) \in L$. Since L is a very long list, this has complexity less than 2^n. Say such a d_L has been found, and it matched the jth element in L. Now, choose the message $d^* \in \mathcal{E}$ such that the number of blocks of $d^* \| d_L$ is j, meaning that the first j blocks in M can be replaced with $d^* \| d_L$ without changing the length of the message. Let M' be this new version of M. Then $H(M') = H(M)$.

The expandable message set can be quite easily produced if the hash function contains fixed points, see Section 2.1, but it is also possible in general by finding collisions between messages of lengths 1 and a for different values of a, and then concatenate messages in different ways similarly to the multi-collision attack described above.

2.2.4 Length-extension Attack

The Merkle-Damgård construction is susceptible to a length-extension attack (Ferguson and Schneier, 2003): Given a hash function H, assume an attacker knows $H(d)$ and the length of d. In the Merkle-Damgård construction he can then select a suffix x, and compute $H(d \| x)$ without knowledge of d. He does this by setting the first part of x to be the padding of d (which he knows since he knows $|d|$), and he is then free to choose the remainder of x.

2.2.5 Discussion on the Generic Attacks

The attacks just described show that it is too simple to build messages with certain properties in the Merkle-Damgård construction. The effect of replacing a part of a message by something else, or prepending or appending some message to another message, is too modest. Moreover, recent short-cut attacks (Biham et al., 2005; Wang et al., 2005; Wang and Yu, 2005) on hash functions such as MD5 and SHA-1 prove that it is not as easy to construct a collision resistant compression function as once believed.

3 POSSIBLE COUNTERMEASURES

In this section we discuss some possible methods for defeating the attacks and weaknesses mentioned in the previous sections.

3.1 Alternatives to Davies-Meyer

There are twelve secure compression function constructions (Preneel et al., 1993) based on a family of 2^n bijections on n bits. For eight of these, including the Davies-Meyer construction, it is relatively easy to find fixed points. One of the remaining four is the "dual mode" to the Davies-Meyer construction

$$h_i = f(h_{i-1}, d_i) = p(d_i, h_{i-1}) + d_i, \qquad (2)$$

attributed to Matyas, Meyer, and Oseas (Preneel, 1993). Note that the problems of (1) stem from the facts that p is invertible when the second argument is fixed, and that the second argument is formed from the message block (alone), and thus the attacker has complete control over this. This is not the case in (2), where the second argument is formed from the chaining variable alone. In this light it may seem strange why (1) is more employed in practice than (2), but this is probably due to a choice of efficiency on behalf of security. One advantage of (1) compared to (2) (seen from the designer's point of view) is that the size of the data block in the former can be made larger than the size of the chaining variable, thus providing faster hashing.

3.2 Extensions and Alternatives to Merkle-Damgård

It is not as easy as it may seem to protect against the generic attacks on the Merkle-Damgård construction. The following is a discussion on what might and what might not work.

3.2.1 3c, A Recent Proposal

A recent proposal named 3C (Gauravaram et al., 2006) continuously updates an additional variable by adding (xoring) to it the chaining variable after each iteration. In the end, this additional variable is converted into a message block, which is appended to the original message. This method might complicate short-cut attacks, but it does not have any effect on the multi-collision attack, since the chaining variables are identical after each iteration.

Since 3C is just Merkle-Damgård with an extra message block derived from all intermediate chaining values, fixed points can be found in the same way as in the standard Merkle-Damgård construction. In 3C, if the fixed point is applied $2k$ times, then for any positive k the additional message block is the same. However, it seems difficult to make use of this fact, since this does not produce infinitely many collisions because of the length padding, and Kelsey and Schneier's second preimage attack is thwarted because it is hard to ensure that the additional message block has the same value as for the target message.

In the following section we analyse more general methods of appending a checksum-like value to the message. The aim is to complicate the multi-collision attack of Joux.

3.2.2 Appending a Checksum

The hash function MD2 is an iterated hash function which differs from constructions of the SHA-family in several ways. One particular distinct feature is the use of a checksum function (RFC 1319, 1992), which computes an additional message block as a function of all other (original) message blocks. The checksum block is appended to the original message. It shall be assumed that the checksum is computed iteratively as follows. Let $d_1, d_2, \ldots, d_s$ be the blocks to be hashed and let $c_1, c_2, \ldots, c_s$ denote the intermediate checksum, such that, $c_1 = C(d_1), c_2 = C(d_1, d_2)$, and $c_s = C(d_1, d_2, \ldots, d_s)$. We shall show that such checksums do not always provide much added security against some of the generic attacks listed above.

Consider first the simple checksum function where one computes the exclusive-or sum of all data blocks. Or more generally, consider checksum functions such that there exist invertible subfunctions $C_1, C_2, \ldots, C_s$ such that

$$c_s = C(d_1, d_2, \ldots, d_s) = \sum_{i=1}^{s} C_i(d_i). \qquad (3)$$

Such checksum functions do not add much protection against the multi-collision attack. Let $\mathcal{C}$ be a collision finder of the compression function, that always returns one-block collisions where for the two messages d_i, d_i' the checksum values $C_i(d_i)$ and $C_i(d_i')$ agree on the first $m-(n/2+\epsilon)$ bits. For positive but small ϵ, this gives enough freedom for $\mathcal{C}$ to actually find such collisions. Choose $t > n/2 + \epsilon$. Find a chain of t collisions, which form an intermediate 2^t-way multi-collision, excluding the checksum. Now choose some value S for the checksum of all blocks – of course, the first $m - (n/2 + \epsilon)$ bits of S are fixed by the messages in the multi-collision. Form two sets of checksums; the checksums of all $2^{t/2}$ combinations of the first $t/2$ blocks, and the value S subtracted the checksums of the last $t/2$ blocks. From the relation (3) one sees that this corresponds to all possible values of $c_{t/2}$ such that S is reached by the last $t/2$ blocks. Since $t > n/2 + \epsilon$ and the first $m - (n/2 + \epsilon)$ bits always match, with good probability there will be a match between the two sets. The running time of the attack which finds (roughly) a $2^{t-n/2}$-collision is the time it takes to find the t collisions, plus the time it takes to compute about $2^{t/2+1}$ checksums (or inversions). If $t = n$, for instance, then a $2^{n/2}-$collision is expected with a total running time of about $(n+2)2^{n/2}$. Note that Joux's attack on a construction without the checksum finds a $2^{n/2}$-collision in time $(n/2)2^{n/2}$.

The checksum function of MD2 is not as simple as the above. There is a rotation of the bytes in a block and applications of a nonlinear S-box, features which make it seemingly impossible to ensure that the final checksum has a number of fixed bits as above – at least when the messages are long enough. Assume, however, that a checksum function similar to the one of MD2 is used to construct a checksum of μ bits, and this checksum is appended to the message (padded, if necessary). Assume that this checksum function is invertible, that is, if $c_s = c(d_1, \ldots, d_s)$, is the checksum function value after processing the sth block, then it should be possible given d_s to compute $c_{s-1} = c(d_1, \ldots, d_{s-1})$, The MD2 checksum function is invertible under this definition. An attacker would choose some value of the checksum, say S. He would then build a, say, 2^t-way multi-collision as usual, but not considering the checksum. Similar to above, he would compute $2^{t/2}$ intermediate checksums of the first $t/2$ blocks of each message, and $2^{t/2}$ intermediate checksums of the last $t/2$ blocks in a backward direction starting from S. These last $2^{t/2}$ checksums are those from which S is reached by the last $2^{t/2}$ blocks. He would then perform a meet-in-

the-middle search for matches between the two sets of checksums. One expects to get multi-collisions with $2^{t-\mu}$ matches. The complexity of a $2^{t-\mu}$-collision attack is roughly the time it takes to find the t collisions on the compression functions, plus the time it takes to compute about $2^{t/2+1}$ checksums.

In this case the complexity of the multi-collision attack depends very much on μ, the size of the output. In the case of MD2, $\mu = n$, so multi-collisions for MD2 can be found considerably faster than by a brute-force attack. However, if μ would be much larger than n, say $\mu = 2n$, then the complexity of the (generic) attacks would be much higher.

These considerations indicate that if one wants to use a checksum to protect against the multi-collision attack, then one will have to select the checksum function carefully. At the very least the checksum value that is appended to the message must be large enough to make the attacks described at least as hard as a brute-force attack, or the checksum function must be non-invertible.

An alternative method is to insert a (simpler) checksum block for every j iterations, with e.g. $j = 8$. This would prevent an attacker from being able to choose t large enough to ensure that the 2^t-way multi-collisions would contain collisions also on the checksum.

The herding attack and the 2nd preimage attack are also complicated by appending or inserting the output of a checksum function. It becomes much more difficult to replace part of a message with something else, or to use just any of a large set of messages as in the herding attack. One would have to select a value for the checksum before the hash value is chosen, and not much freedom is left for finding the linking message in the end.

The length-extension attack is complicated by the fact that the adversary does not know the checksum of the original message.

The drawbacks of the checksum approach are that it requires both time and memory to compute and store the checksum. What's more, the resulting message length increases, and so the compression function must be invoked an additional number of times. However, the checksum function can be constructed such that it is much faster than the compression function, and the rather limited number of additional bits of message do not affect the total running time by much.

3.2.3 A Wide-pipe Method

As first suggested by Lucks (Lucks, 2004), the security of the Merkle-Damgård construction might be regained by choosing a compression function with a larger output size than the final output of the hash function. This would make each regular collision of the multi-collision attack more time-consuming, assuming the quickest method to find collisions of the compression function is the birthday attack. By doubling the internal width of the hash function, the complexity of the multi-collision attack becomes greater than the complexity of a brute-force multi-collision attack.

This method also complicates the other attacks mentioned. The herding attack and the 2nd preimage attack are complicated in that each collision is more difficult to find. The length-extension attack is complicated in that the adversary does not know the final output of the compression function, because this is not the same as the output of the hash function.

Again, the most significant drawback of this method is that it slows down the entire operation – assuming that a larger compression function is slower. In any case, an output transformation from the final output of the compression function, to the output of the hash function, is needed. This could be a narrow version of the compression function or something else, but this is an additional potential weak point of the hash function.

4 PROPOSALS FOR ENHANCED SECURITY

This section gives a proposal for a variant of the Merkle-Damgård construction. This variant, which bears some resemblance to a proposal by Rivest (Rivest, 2005), is constructed with the SHA-family in mind and it has some attractive properties, as we shall see.

Assume the existence of a (secure) compression function

$$h : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n.$$

Define a hash function H as follows.

- Let d be the data to be hashed, and append a '1'-bit to d. Now append enough '0'-bits to make the length of d a multiple of m. Now $d = d_1 \| d_2 \| \cdots \| d_s$, where $|d_i| = m$, $1 \leq i \leq s$. It is specifically required that $s < 2^{n/2}$.
- Let iv_1 and iv_2 be given (fixed) n-bit initial values, and let $h_0 = \text{iv}_1$.
- Define

$$h_i = h(\text{iv}_1 + i, h_{i-1}, d_i) \quad \text{for } 1 \leq i \leq s$$

- Define $H(d) = h(\text{iv}_2, h_s, s)$ to be the output of the hash function.

The value $\text{iv}_1 + i$ is to be computed modulo 2^n. The values of iv_1 and iv_2 should be chosen such that $\text{iv}_2 \neq \text{iv}_1 + i$ for any admissible value of i. Note

that this hash function can hash data strings of up to a number of blocks equal to the minimum of $2^m - 1$ and $2^{n/2} - 1$. If $m < n/2$ it is easy to extend the above construction to allow for a number of blocks up to $2^{n/2} - 1$. Simply split s into a pre-determined fixed number of blocks, say t, each of m bits for $s = s_1, s_2, \ldots, s_t$. Then modify the last step of above to $x_j = h(\text{iv}_2, x_{j-1}, s_j)$, for $1 \leq j \leq t$, where $x_0 = h_s$ and define $H(d) = x_t$.

Theorem 2. *Let H be a hash function as defined above. Let iv_1 and iv_2 be given. Then any collision on H implies a collision on h where the first arguments are identical. In other words, the existence of two distinct strings d and d' such that $H(d) = H(d')$ implies the existence of two distinct triples (x, y, z) and (x, y', z') such that $h(x, y, z) = h(x, y', z')$.*

Proof: Assume a collision for H, that is, $d \neq d'$, such that $H(d) = H(d')$. Let s and s' denote the number of blocks in d and d' after the padding bits. If $s \neq s'$, the result follows from $h(\text{iv}_2, h_s, s) = h(\text{iv}_2, h'_s, s')$. Assume next that $s = s'$. Then if $h_s \neq h'_s$, the result follows again. If $h_s = h'_s$, then it follows that for some j, $1 \leq j < s$, one gets $h(\text{iv}_1 + j, h_{j-1}, d_j) = h(\text{iv}_1 + j, h'_{j-1}, d'_j)$ and $(h_{j-1}, d_j) \neq (h'_{j-1}, d'_j)$. ∎

Theorem 2 shows that if h is a compression function resistant to collisions where the first arguments are identical, then H is collision resistant.

4.1 Application to the SHA-family Constructions

In the following we show how the members of the SHA-family can be turned into constructions of the above type. Given a hash function H with compression function $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$, where $m > n$, of the form

$$f(h_{i-1}, d_i) = g(h_{i-1}, d_i) + h_{i-1},$$

cf. Section 2.1. Consider the following variant $\tilde{H}$ of H. Let the compression function be

$$h : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^{m-n} \to \{0,1\}^n.$$

- Let $d = d_1 \| \cdots \| d_s$ be the data to be hashed (including padding bits), where $|d_i| = m - n$ for $1 \leq i \leq s$, and $s < 2^{n/2}$.
- Let iv_1, iv_2 and h_0 be given n-bit initial values (these are chosen once and for all in the specification of the hash function, and such that $\text{iv}_1 + i \neq \text{iv}_2$ for any admissible i).
- Define $h_i = h(\text{iv}_1 + i, h_{i-1}, d_i) = g(\text{iv}_1 + i, (d_i \| h_{i-1})) + h_{i-1}$ for $1 \leq i \leq s$.
- Define $\tilde{H}(d) = h(\text{iv}_2, h_s, s) = g(\text{iv}_2, (s \| h_s)) + h_s$.

If the compression function of SHA-n is resistant to collisions where the first input is identical for the two messages, then the construction above using SHA-n is a collision resistant hash function.

Note that a pseudo-preimage for the compression function can be found in time $2^{n/2}$. Given h_i, choose arbitrary values of h_{i-1} and d_i, and invert g. This requires an expected $2^{n/2}$ work, since $2^{n/2}$ different values of i are admissible, and for each d_i, a random n-bit value corresponding to the first argument of g is found (in assumed constant time). Here, it is assumed that the fastest method to find an admissible i is by brute force. A pseudo-preimage is, however, not immediately useful since the attacker has no control over i, and it seems difficult to exploit such findings to find preimages of the hash function.

Since inverting g takes time $2^{n/2}$, this is also the time it takes to find a fixed point of the compression function. Therefore it seems that the 2nd preimage attack of Kelsey and Schneier using fixed points to produce expandable messages is no faster than a brute force attack. Furthermore, the use of a counter in the first argument means that the generic method of producing expandable messages is also not applicable: a one-block message cannot be replaced by an a-block message without changing the input to the following application of the compression function.

The length-extension attack can no longer be carried out. The padding block is processed with iv_2 as the first argument to the compression function, and when the attacker selects this block as part of the extension to the original message, $\text{iv}_1 + i$ for some i will be used as the first argument. By definition, these two values cannot be the same.

The reader may have observed that in the construction above we have not introduced any dedicated measures to try to protect against the multi-collision attack, nor against the herding attack. However, these attacks evidently require at least one collision on the compression function. Thus if the size of the compression function is large enough to make the birthday attack computationally infeasible and if one's aim is to construct a compression function resistant to collisions then this also protects against the multi-collision attacks. Nonetheless, the multi-collision attacks on the iterated hash function constructions do illustrate properties which are not inherent in a "random hashing". If one wishes a construction where the complexity of the multi-collision and the herding attacks is higher than for the birthday attack, we list two possible ways:

- Truncation of the final hash value.
- The addition of checksums.

As an example of the first item take SHA-512. Here the complexity of a birthday attack is 2^{256}, thus today, one may safely truncate the final output to less

than 512 bits, say, 320 bits. Here the birthday collision attack has complexity 2^{160}, whereas a collision of a compression function (except for the final application in a hashing) requires about 2^{256} operations. For the other members of the SHA-family we do not recommend this measure.

As for the second item and applications which employ members of the SHA-family in the proposed construction. Here we'd recommend to append to the original message the values of two checksums, whose outputs each has the size of one message block. These two checksums must be different in such a way as to avoid that by fixing certain bits in the message blocks one also fixes certain bits in the checksums.

4.2 Performance

There is a slowdown when a hash function uses a compression function in our new mode of operation as opposed to the Merkle-Damgård mode. If the compression function takes an $(n+m)$-bit input and produces an n-bit output, then in the traditional mode of operation, m bits of message are processed per iteration of the compression function, whereas in our new mode of operation $m-n$ bits are processed. This slows down the whole operation by a factor $m/(m-n)$. In SHA-1, $n = 160$ and $m = 512$, so the slowdown when using the compression function of SHA-1 in our new construction is by a factor of about 1.5. In SHA-256, $n = 256$ and $m = 512$, and so here the slowdown is by a factor of about 2.0 which is also the case for SHA-512.

4.3 Resistance to Short-cut Attacks

It is clear that the above proposed construction when applied to members of the SHA-family gives the attacker less freedom, since there are fewer bits that he can choose. Currently, the best collision attack on SHA-1 is the one discovered by Wang et al. (Wang et al., 2005). This attack produces colliding messages each formed by two blocks, with a near-collision after the first block. Such an approach does not apply to our construction used with SHA-1, because of the use of the constant iv_1 in the first argument of h.

The attacks of Wang rely heavily on message modifications (Wang and Yu, 2005; Wang et al., 2005), which do not leave much freedom for the attacker to choose message bits. In our construction, 160 bits of the message block in SHA-1 are reserved for the bits of the chaining variable, which makes life harder for the attacker. But clearly this is not a strong argument for our construction. The security arguments for an iterated hash function relies on the collision resistance of the compression function. If a collision is found for the latter, then there are no longer good arguments to rely on the security of the hash function. However, seen in the light of the recent developments in hash function cryptanalysis, including the SHA-1 attacks, there are good reasons to limit the freedom of the attacker in future designs. This is a central idea in our construction.

5 CONCLUSION

We have discussed some generic attacks on the Merkle-Damgård construction and possible countermeasures to these. It turns out that some of these attacks, notably the multi-collision and the herding attacks, seem to be more difficult to protect against than one might expect.

A new method for iterated hash function construction has been proposed. Given an n-bit compression function resistant to collisions where n bits of the two inputs to the compression function are identical, our construction ensures a completely collision resistant hash function. In its bare form our construction does not give added protection against the multi-collision nor the herding attacks, but suggestions were given for possible ways of efficiently thwarting these attacks as well.

ACKNOWLEDGEMENTS

The authors would like to thank Praveen Gauravaram for helpful discussions.

REFERENCES

Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., and Jalby, W. (2005). Collisions of SHA-0 and Reduced SHA-1. In (Cramer, 2005), pages 36–57.

Brassard, G., editor (1990). *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*. Springer.

Cramer, R., editor (2005). *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer.

Damgård, I. (1989). A Design Principle for Hash Functions. In (Brassard, 1990), pages 416–427.

Ferguson, N. and Schneier, B. (2003). *Practical Cryptography*. Wiley Publishing.

FIPS180b (1995). FIPS 180-1, Secure Hash Standard. Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, Virginia. Supersedes FIPS 180.

FIPS180c (2002). FIPS 180-2, Secure Hash Standard. Federal Information Processing Standards Publication 180-2, U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, Virginia. Supersedes FIPS 180 and FIPS 180-1.

Gauravaram, P., Millan, W., Dawson, E., and Viswanathan, K. (2006). Constructing Secure Hash Functions by Enhancing Merkle-Damgård construction. To be published in the proceedings of Australasian Conference on Information Security and Privacy (ACISP, 2006). The paper is available at `http://www.isi.qut.edu.au/people/subramap/`.

Handschuh, H., Knudsen, L., and Robshaw, M. (2001). Analysis of SHA-1 in encryption mode. In Naccache, D., editor, *Topics in Cryptology – CT-RSA 2001*, *Lecture Notes in Computer Science 2020*, pages 70–83. Springer Verlag.

Joux, A. (2004). Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In Franklin, M. K., editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316. Springer.

Kelsey, J. and Kohno, T. (2006). Herding Hash Functions and the Nostradamus Attack. In Vaudenay, S., editor, *EUROCRYPT*, Lecture Notes in Computer Science. Springer. To appear.

Kelsey, J. and Schneier, B. (2005). Second Preimages on n-bit Hash Functions for Much Less than 2^n Work. In (Cramer, 2005), pages 474–490.

Lucks, S. (2004). Design Principles for Iterated Hash Functions. Cryptology ePrint Archive, Report 2004/253. `http://eprint.iacr.org/`.

Merkle, R. C. (1989). One Way Hash Functions and DES. In (Brassard, 1990), pages 428–446.

Preneel, B. (1993). *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven.

Preneel, B., Govaerts, R., and Vandewalle, J. (1993). Hash Functions Based on Block Ciphers: A Synthetic Approach. In Stinson, D. R., editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer.

RFC 1319 (1992). RFC 1319, The MD2 Message-Digest Algorithm. Internet Request for Comments 1319, B. Kaliski.

Rivest, R. L. (2005). Abelian square-free dithering for iterated hash functions. Presented at the NIST Cryptographic Hash Workshop, November 2005, and retrieved from `http://theory.lcs.mit.edu/~rivest/`.

Wang, X., Yin, Y. L., and Yu, H. (2005). Finding Collisions in the Full SHA-1. In Shoup, V., editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer.

Wang, X. and Yu, H. (2005). How to Break MD5 and Other Hash Functions. In (Cramer, 2005), pages 19–35.