SOME ATTACKS ON CRYPTOGRAPHIC HASH FUNCTIONS

C. Krishna Kumar [1], Dr. C. Suyambulingom[2]
1, Sathyabama University, Chennai, India,
2 Professor, Dept. of Mathematics, TAU,
Coimbatore, India
E-Mail: kkumarmalar@yahoo.com

Abstract

Hash functions are used in various applications which require specific properties. Therefore, these cryptographic tools are designed in order to satisfy the desired properties The methods for analyzing hash functions, or attacks shortly, can be classified in various types according to the main parameters used in the attack. For example some attacks depend only on the hash size while other attacks may also depend on chaining value or compression function. At this point, one can categorize attacks, in a more general fashion, as generic attacks and specific attacks. Generic attacks are general attacks that are mostly applicable to numerous hash functions. Specific attacks, however, are cryptanalysis methods for specific hash functions and applicable to very limited number of algorithms. Specific attacks are out of the scope of this thesis and generic attacks will be mentioned in this section.

1. Birthday Attack

Birthday attack is the basic tool to find collisions for any hash function. The attack is based on the birthday problem (or birthday paradox) which is about the minimum number of people so that the probability of at least two people having the same birthday is greater than 1/2. To bind the problem to hash functions, one can call the presence of two people having the same birthday a collision.The key point in birthday attack is that the attacker, is looking for any collision rather than a specific collision. If attacker was looking for a specific collision among N people, say at least one person having the same birthday with her, the probability will be

$PCollision(N) = 1 - (364/365)^N$

For this probability being greater than 1/2, N should be greater than 254. Applying this idea to hash functions, attacker is trying to find a message which has a hash value with a specific message by exhaustively searching all possible messages which would be a second preimage search. However, if attacker just tries to find any collision among the N birthdays the probability Becomes $PCollision(N) = 1 - (365/365)(364/365)(363/365)….(365-N+1/365)^N$

Here solving for N, it can be seen that N = 23 people are enough to have a match in the birthdays with probability greater than 1/2.

2. Correcting Block Attack

Correcting block attack is the most trivial second preimage tool which can also be used to find collisions. The idea is, roughly, for a given message M1 of t-blocks and corresponding hash value H(M1), producing a message M2 of k-blocks which is shorter than M1 and then trying to find message blocks $Y = m^1c, \cdots, m^{t-k}c$ such that $H(M_1) = H(M_2||Y)$. In general, the attack is applied by correcting the last message block before padding and called correcting last block attack. In this case, attacker produces a message M2 of length t−1 blocks. Let ht and h^{t-1} be, respectively, the final chaining values of messages M1 and M2 before processing padding blocks. Then the attacker searches for a message block mc such that f (h t −1,mc) = ht where f is the

compression function of the hash algorithm. This way H(M2||mc) will be equal to H(M1) and she will get a second preimage for M1. The choice of message block to be corrected has no restriction and can be any message block but the first. If attacker chooses ith message block to correct to find a second preimage, message blocks of M2 after this message block should be equal to message blocks of M1 after ith message blocks. To find a collision, attacker randomly chooses two messages of the same length M1 and M2. Then, he searches for message blocks mc and m csuch that H(M1||mc) = H(M2||mc) and gets a collision pair.The complexity of finding such a correcting block is equal to exhaustive collision search for most of the modern hash functions and it is O(2n). However, some hash functions are easy to manipulate and this process can be less complicated. Especially hash functions based on modular arithmetic are vulnerable to correcting block attack.

3. Meet in the Middle Attack

The meet in the middle attack (MiMA) is a variant of birthday attack to find second preimages. In the birthday attack, attacker is comparing the hash values while in MiMA, she compares the internal chaining values. The advantage of MiMA over birthday attack is that MiMA enables attacker to construct a message with a desired hash value, ie., a second preimage. However, MiMA is applicable to hash functions with invertible compression functions where birthday attack can be applied to any hash function. In other words, to apply MiMA, attacker, knowing hi+1, should be able to find a pair (hi,mi) st f (hi,mi) = hi + 1. General idea of MiMA is going backwards from a hash value and going forwards from IV of the algorithm using some bogus message blocks, and then at a predefined meeting point trying to match them. In [66], Nishimura and Sibuya defined three models for MiMA. In the first model, model A, attacker who knows M and H(M), divides the bogus message M into two parts. She chooses a chaining value as meeting point and starting from IV, generates k variations of the first part up to the meeting point.

4. Length Extension Attack

Length extension attack can be applied to the iterated hash functions by appending message blocks to the original or padded message. The aim of this attack is producing a hash value which is related to or contains a part of a message M without fully knowing it. The length extension attacks can be categorized in two as Type A and Type B attacks. In Type A attack, the hash function is assumed to use no MD-strengthening. For this type of attack consider two messages, M1 = m0,m1, . . . ,mt−1 and M2 = m0,m1, . . . ,mt−1,mt which are identical in the first t blocks. Then, when calculating the hash value of M2, H(M1) appears as the chaining value which is input to the final iteration. Therefore H(M2) = f (H(M1),mt) where f is the compression function of H. This enables the attacker, without full knowledge of M1, to produce the hash value of a message M2 which is related with M1. One example of the attack is given in [69]. In an authentication protocol, which uses an iterated hash function without output transformation, two parties A and B share a secret X and the authentication protocol is as follows:

• A, generates some message M and sends M, H(X||M) to B

• B, knowing X, compares H(X||M) with the received data and authenticates A

Figure 4.4: The Authentication Protocol

Here an attacker who knows M and H(X||M) can impersonate A to B with extension

attack. The attacker generates arbitrary message blocks m0,m1, . . . ,mk−1 and iterates hi = f (hi−1,mi−1) for i = 1, 2, .., k with h0 = H(X||M). The result, hk, is the hash value of M = X||M||m0,m1, . . . ,mk−1. Then the attacker sends M and hk to B. Party B, verifying the hash value, authenticates attacker as A.

Fixed Point Attack A fixed point for the hash function H is a pair (h,m) such that f (h,m) = h, ie, the chaining value h remains the same after iteration with message block m. Fixed points are not just the chaining values or the message blocks: h which is a fixed point for the message block m will not form a fixed point pair with another message block, or vice versa. Fixed points are used to attack iterated hash functions. Let the pair (hi,mi) be a fixed point pair for the compression function f of H. Consider the message M = m0,m1, . . . ,mi, . . . ,mt−1 with chaining values IV = h0, h1, . . . , hi, . . . , ht appear when hashing M.

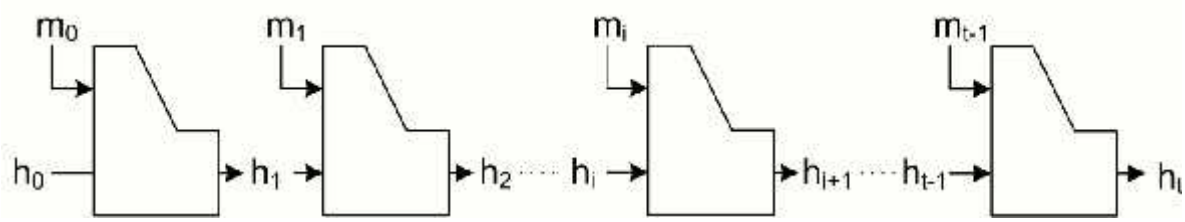

Figure1.11: Hashing Process of M

Attacker can construct another message M^1 = m0,m1, . . . ,mi−1,mi,mi, . . . ,mt−1. which produces the chaining values IV = h0, h1, . . . , hi−1hi, hi, hi+1, . . . , ht.

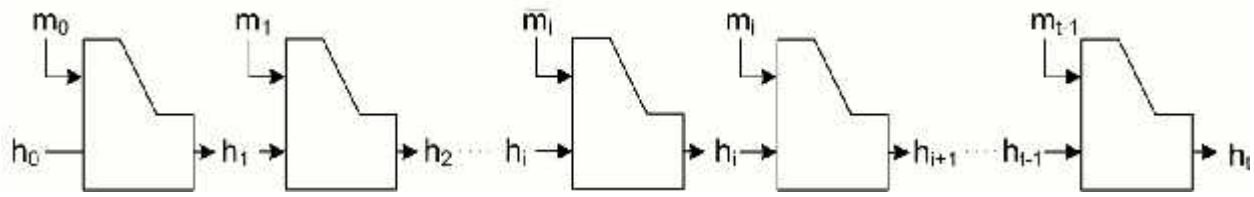

Figure 1.12: Hashing Process of M^1

Since (hi,mi) forms a fixed point pair, insertion of mi to the iterations where hi is the input, will not change the final chaining value. So the attacker can insert as many mi message blocks as she needed without changing the hash value to find a second preimage. However, the problem with this attack is the length padding. As she inserts message blocks to produce a second preimage, the message M^1 will be longer than the original message M. Therefore the length paddings of two messages will be different and the hash values will not be the same.

5. Long Message Attack

The long message attack is a second preimage attack using long messages for iterative hash functions. Attack is applicable to messages of any length but it is more efficient when the original message is very long. Assume the attacker is trying to find a second preimage for the message M = m0m1m2 . . .ml−1 which produces the chaining values h0 = IV, h1, . . . , hl − 1, hl

= H(M) when hashing. 42 Attacker produces a prefix M = m0m1 . . .mk−1 with final chaining value f (hk−1,mk−1) = hk. If hk is equal to one of the chaining values h1, h2, . . . , ht−1, the attacker can bind the message blocks of the original message after the matching chaining value to M. Let hk = ha, then M will be M = m0m1 . . .mk−1mama+1 . . .ml−1 and will have the same hash value with M. If hk is not equal to any of the chaining values, then attacker tries to find a linking message mlink to produce another chaining value hk+1 which is equal to one of the chaining values. Since there are l − 1 chaining values appearing in hashing process of M, assumed to be all distinct, hk, or hk+1, can be equal to one of these chaining values with probability l−1 2n . Letting l = 2b, the cost of a match will be 2n−b hash function computations. The final chaining values of M and M are the same. However, since length padding is processed at all modern hash functions , hash values of these messages can be different depending of their lengths. Let the matching chaining values of these messages be ha = hk. Then there are three situations:

• If k < a, then M is shorter then M. Then the attacker can use the methods used in the fixed point attack, in Section 4.6, proposed by Dean [53] to extend the message M.

• If k = a, then the lengths of the messages will be equal. Therefore the hash values H(M) and H(M) will be equal.

• If k > a, then M is longer than the original message M. This time attacker should use the tricks in the fixed point attack to shrink the message to the same length with M.

Moreover, Kelsey and Schneier developed an attack, that will be explained in the next section, which can be applied using the long message attack.

6. Kelsey & Schneier's Long Message Attack

In 2005, Kelsey and Schneier proposed a new second preimage attack which is essentially similar to Joux multi-collision attack and integrated with the long message attack [55]. The attack can be combined with Dean's fixed point idea for hash functions that are easy to find fixed points. Attack is basically, as Joux's attack, depends on finding successive collisions. The main 43 difference of this attack from the Joux multi-collision is, instead of finding colliding pairs of one block messages, attacker searches for pairs of collisions of different lengths. This way she can construct a message having a fixed hash value but that can vary in length. Consider this message as a single elastic message. This elastic message is called an expandable message and an expandable message that can take any length between a and b blocks is called an (a, b) − expandable message. Finding collisions for different length messages can be achieved in two ways. First method is using the fixed points. In this method, attacker generates 2n/2 fixed point pairs (hf i ,mf i) and 2n/2 chaining values h1i that can be reached from IV using message block m0i . With a high probability there will be a match between hf 's and h1's. Let h1i = hfj , then the messages m0i and m0i ||mfj will produce the same final chaining values. Attacker can also append as many mfj blocks to the second message as she wants. This method is only applicable to hash functions which are easy to find fixed points.

7. Herding Attack

Kelsey and Kohno developed an attack on Merkle-Damgard hash functions in [74], where the attacker first reveals the hash value and then produces a message which

has the revealed hash value. Since the attacker produces the message after finding its hash value, she can prove prior knowledge of any information. Therefore, the attack is also called Nostradamus attack. An example to clarify the attack can be as follows: Attacker claims that she knows the result of a match which will be played soon and provides a hash value which she declares as the hash value of the result of the match. After the match, she publishes a message containing the result of the match which has the previously declared hash value. This attack has two phases: first the attacker constructs a diamond structure which contains colliding chaining values. These chaining values reduce to a single chaining value ht and attacker releases ht.

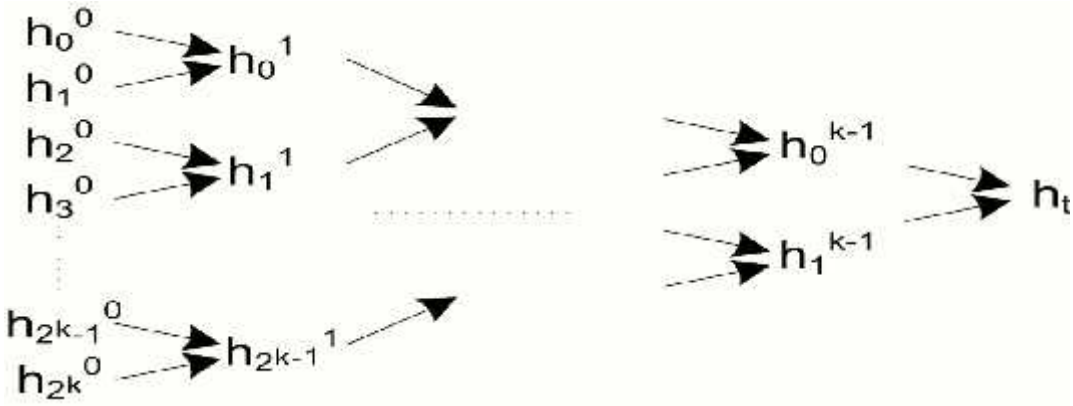

Figure 1.13: Diamond Structure

In Figure 1.13 a diamond structure is depicted where each arrow denotes a message block which are not necessarily distinct. After learning the result of the match, which will be called the prefix P, she finds a linking message mlink that links the final chaining value of the prefix to the first column of the diamond structure. The diamond structure is consisting of colliding messages. To construct a diamond structure of width k one should find 2k multi-collisions. The idea of diamond structure is similar to Joux multi-collision however there are two main differences:

1. The diamond structure allows 2k choices for the first message block and the rest of the messages are determined by the first block while Joux attack provides two message block choices for each iteration.
2. Diamond structure contains $2k+1 - 2$ chaining values. Joux $2k-$multi-collision has k chaining values. The first difference makes finding a linking message more efficient and the second enables the attacker to apply herding attack to short suffixes.

8. Slide Attack

Slide attack is a cryptanalysis tool for block ciphers with weak key schedule, which was introduced by Wagner and Biryakov in [76]. However, in [77] Gorski, Lucks and Peyrin showed that the attack is also a threat for stream based and sponge hash functions. By applying slide attack to hash functions one can distinguish the algorithm from a random oracle and mount an extension attack similar to the length extension attack for MD hash functions in Section 4.4.Slide attack is applicable to block ciphers which have a weak or periodic key schedule. These block ciphers can be divided into equivalent

permutations according to the round keys. For example, let the key size of a 16 round block cipher E is 128 bits and ith round key kiisproduced by rotating the main key left by i 32. Then the round keys will repeat after 4 rounds and will be equal at each four rounds. Let F be the permutation equal to the first four rounds of E. Then, one can consider E as E(P) = F ◦ F ◦ F ◦ F(P). If the corresponding ciphertexts C1 and C2 of plaintexts P1 and P2, with P2 = F(P1), satisfies C2 = F(C1) then (P1, P2) is called a slid pair. Using the slid pairs attacker can retrieve information about the key and distinguish the cipher from a random oracle. In hash function case, one can consider the blank rounds at the end of stream and sponge based hash functions as identical permutations which does not take any input apart from the chaining value and applies a fixed permutation on it. So, parallel to these consideration, the definition of a slid pair will change. Two messages M1 and M2 are called a slid pair if S 2 = B(S 1) where B(·) denotes a blank round and S 1 and S 2 are the final states appear before the output transformation, when hashing these messages. There is no constraints on the messages as in block cipher case. If the state after processing the first blank round of M1 is equal to the state before the blank rounds of M2, (M1, M2) will be a slid pair. Assume that the attacker is trying to attack the protocol as in Figure 4.4 of Section 4.4. Stream and sponge hash functions apply an output transformation to the state (or states) after blank rounds. Therefore, in such a protocol, attacker who knows H(K||M) and M cannot recover the final state and append new message blocks directly. For example, in Grindahl [78], the final state is truncated according to the hash size. So, in order to extend the message attacker needs to find the full final state. Slide attack enables the attacker to find the final state with lower complexity. In this section the attack is applied on Grindahl2 hash function. The specifications of Grindahl- 512 can be summarized as follows

• Message M is padded and divided into 64-bit blocks m0m1 . . . .

• An initial state is constructed as a 8x13 matrix whose entries are all zero.

• ith message block is replaced with the first column of the ith state. Then some permutations are applied to the overwritten state.

• 8 blank rounds, which are consisting of permutations only, are applied to the state after all the message blocks are input. In these blank rounds state is neither inserted any message nor truncated.

• The first 8 columns of the state after blank rounds are taken as the output. In other words the final 5 columns are truncated.

9. Rebound Attack

Rebound attack [79] is a differential collision method for block cipher based hash functions. The aim of the attack is, given two messages M1 and M2 such that M1 M2 = Δ, to cancel the difference Δ after some number of iterations or some rounds of encryption in the compression function. Attack divides the block cipher of the compression function into three subciphers as Efw ◦ Ein ◦ Ebw and proceeds in two phases: the inbound phase and the outbound phase. In the inbound phase attacker tries to find a match in the middle using the degrees of freedom in Ein subcipher. Then going backwards and forwards through the matching pairs, tries to get a zero difference at the end, which is produced by the Δ difference. This phase is called the outbound phase and processed through Efw and Ebw. The inbound phase produces tarting points(states) for outbound phase, therefore, inbound phase should be repeated as many times as needed according to the probability of the outbound phase. Despite the attack

can be applied to any block cipher based hash function, it has been applied to, only, hash functions, like Whirlpool[80], Maelstrom [81], and Grostl [82], which use Rijndael [83] building blocks. In the rest of this section, rebound attack will be described on Whirlpool hash function.

References

1.William Stallings, "Cryptography and Network Security", Prentice Hall, 4th Edition, 2005.

2 Andreeva E (2010) Domain Extenders for Cryptographic Hash Functions. Ph.D. thesis, Katholieke Universiteit Leuven.

3. Barkan E, Biham E & Shamir A (2006) Rigorous bounds on cryptanalytic time /memory tradeoffs. Proc. 26th Annual International Cryptology Conference – CRYPTO 2006, 1–21.

4. Bellare M & Ristov T (2008) Hash functions from sigma protocols and improvements to VSH. Proc. 14th International Conference on the Theory and Application of Cryptology and Information Security – ASIACRYPT 2008, 125–142.

5. Blake IF & Shparlinski IE (2007) Statistical distribution and collisions of the VSH. Journal of Mathematical Cryptology 1(4): 329–349.

6 Gauravaram P & Kelsey J (2007) Cryptanalysis of a class of cryptographic hash functions. Cryptology ePrint Archive, Report 2007/277. URL: http://eprint.iacr.org/. Cited 2012/08/24.

7. Klima V (2005) Finding MD5 collisions on a notebook PC using multi-message modifications. Cryptology ePrint Archive, Report 2005/102. URL: http://eprint.iacr.org/. Cited 2012/08/24.

8. Th. Berson, "Differential cryptanalysis mod 232 with applications to MD5," Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp.1–80.

9. C. Besnard and J. Martin, "DABO: proposed additional message authentication algorithms for ISO 8731," preprint, 1992.

10. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," Journal of Cryptology, Vol. 4, No. 1, 1991, pp. 3–72.

11. E. Biham and A. Shamir, "Differential cryptanalysis of Feal and N-hash," Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 1–16.

