Re-Visiting HAIFA and why you should visit too

Orr Dunkelman

Département d'Informatique
École Normale supérieure
and
France Telecom Chaire

3rd of June 2008

Joint work with Eli Biham, Charles Bouillaguet, Pierre-Alain Fouque, Sébastian Zimmer.

Outline

Outline

The Merkle-Damgård Construction

- Presented by Merkle and Damgård independently as an answer to the following problem:
  - Given a compression function $f : \{0,1\}^{m_c} \times \{0,1\}^{n} \rightarrow \{0,1\}^{m_c}$, how would you generate a hash function $H_f : \{0,1\}^{*} \rightarrow \{0,1\}^{m}$.

The Merkle-Damgård Construction

- Presented by Merkle and Damgård independently as an answer to the following problem:
  - Given a compression function $f : \{0,1\}^{m_c} \times \{0,1\}^{n} \rightarrow \{0,1\}^{m_c}$, how would you generate a hash function $H_f : \{0,1\}^{*} \rightarrow \{0,1\}^{m}$.
- The solution is as follows:
  1. Pad the message M to a multiple of b (with 1, and many 0's as needed and the length of the message).
  2. Divided the padded message into l blocks $m_1 m_2 \ldots m_l$.
  3. Set $h_0 = IV$.
  4. For $i = 1$ to l, do $h_i = f(h_{i-1}, m_i)$.
  5. Output h_l (or some function of it).

The Security of the Merkle-Damgård Construction

- Finding a collision in H_f means finding a collision in f.
- Thus, if f is collision-resistant, so is H_f.

The Security of the Merkle-Damgård Construction

- Finding a collision in H_f means finding a collision in f.
- Thus, if f is collision-resistant, so is H_f.
- Also, finding a second preimage in H_f means finding collision in f.

The Security of the Merkle-Damgård Construction

- Finding a collision in H_f means finding a collision in f.
- Thus, if f is collision-resistant, so is H_f.

- Also, finding a second preimage in H_f means finding collision in f.
- The same is true for finding a preimage (because you can use it to find a second preimage).

The Security of the Merkle-Damgård Construction

- Finding a collision in H_f means finding a collision in f.
- Thus, if f is collision-resistant, so is H_f.

- Also, finding a second preimage in H_f means finding collision in f.
- The same is true for finding a preimage (because you can use it to find a second preimage).

To conclude, if f is collision resistant (i.e., it takes $O(2^{m_c/2})$ invocations to find a collision), then H_f is collision resistant and (second) preimage resistant with security level of $O(2^{m_c/2})$.

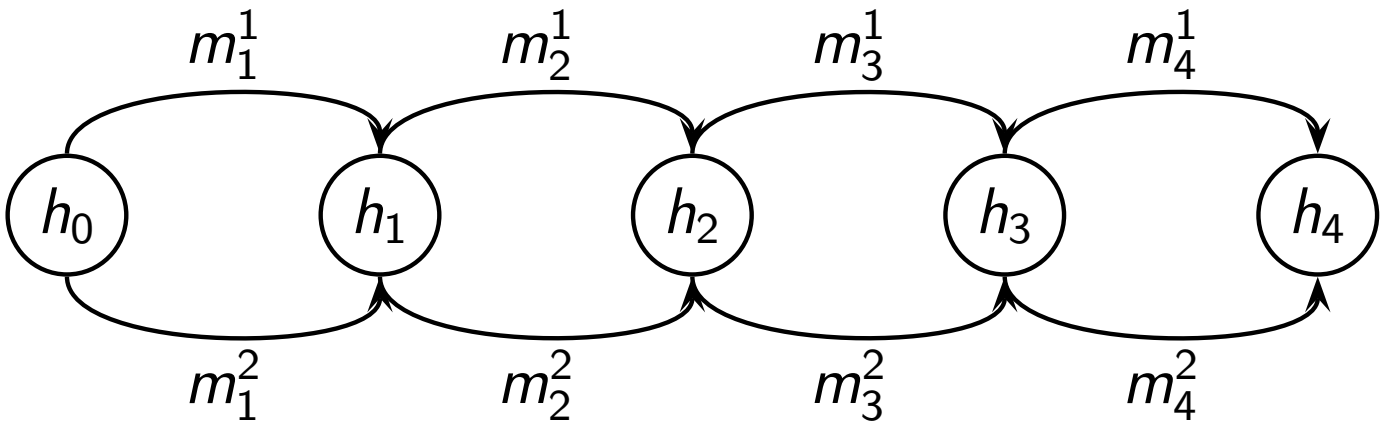
Attacks against the Merkle-Damgård Construction

Let m be the hash size, and m_c be the chaining value size. Let l be the number of blocks of a given message M, and let k be the number of given messages (for one-of-many attacks).

- Easily invertible compression functions (the backward attack) — $2^{m_c/2}$ (late 1970's)
- One-of-many preimage attacks, one-of-many second preimage attacks — 2^{m-k}, $2^{m/2}$ (Merkle 1979)

Generic Attacks against MD Construction (cont.)

- Multi-collision — $O(2^{m_c/2})$ (Joux, 2004)

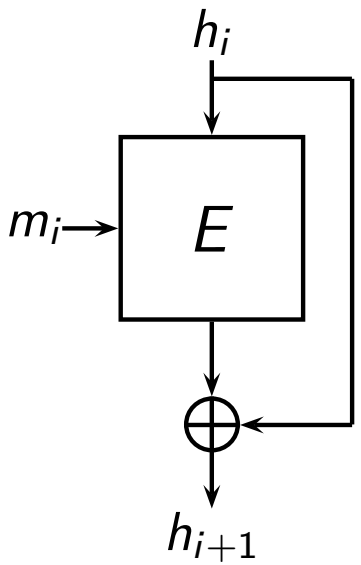


Finding 2^t collisions takes $O(t \cdot 2^{m_c/2})$ (ideally we would expect $O(2^{\frac{2^t-1}{2^t} \cdot m_c})$).

Generic Attacks against MD Construction (cont.)

- Fixpoint and second preimage attacks (long messages) — 2^{m_c-l}, $O(2^{m_c/2})$ (Dean, 1999) [mainly for Davies-Meyer functions]

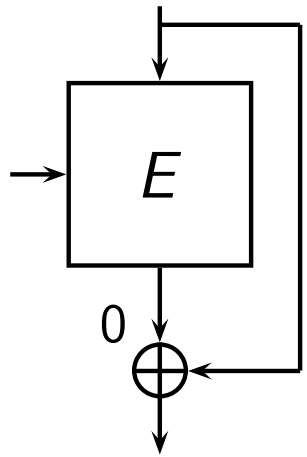
- Find $O(2^{m_c/2})$ fix-points denoted by $A = (h, m)$.
- Select $O(2^{m_c/2})$ single blocks and compute $B = (C_{MD}(IV, \tilde{m}), \tilde{m})$.
- Find a collision between A and B.
- Voilà — an **expandable message** $\tilde{m}||m^t$ for all t lead to the same chaining value h.



Generic Attacks against MD Construction (cont.)

- Fixpoint and second preimage attacks (long messages) — $2^{m_c - l}$, $O(2^{m_c/2})$ (Dean, 1999) [mainly for Davies-Meyer functions]

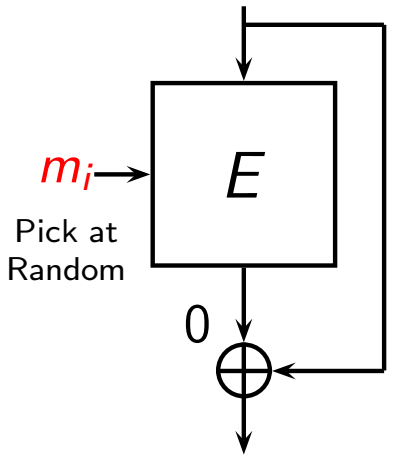
- Find $O(2^{m_c/2})$ fix-points denoted by $A = (h, m)$.
- Select $O(2^{m_c/2})$ single blocks and compute $B = (C_{MD}(IV, \tilde{m}), \tilde{m})$.
- Find a collision between A and B.
- Voilà — an **expandable message** $\tilde{m}||m^t$ for all t lead to the same chaining value h.



Generic Attacks against MD Construction (cont.)

- Fixpoint and second preimage attacks (long messages) — 2^{m_c-l}, $O(2^{m_c/2})$ (Dean, 1999) [mainly for Davies-Meyer functions]

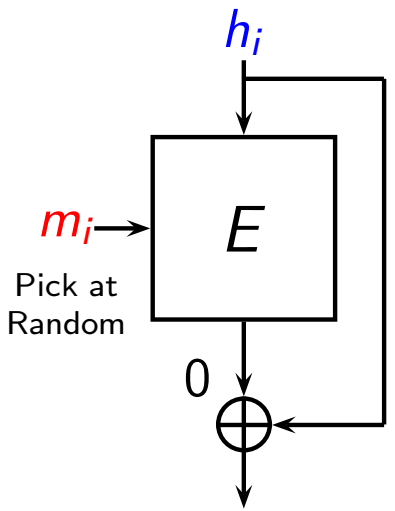
- Find $O(2^{m_c/2})$ fix-points denoted by $A = (h, m)$.
- Select $O(2^{m_c/2})$ single blocks and compute $B = (C_{MD}(IV, \tilde{m}), \tilde{m})$.
- Find a collision between A and B.
- Voilà — an **expandable message** $\tilde{m}||m^t$ for all t lead to the same chaining value h.



Generic Attacks against MD Construction (cont.)

- Fixpoint and second preimage attacks (long messages) — 2^{m_c-l}, $O(2^{m_c/2})$ (Dean, 1999) [mainly for Davies-Meyer functions]

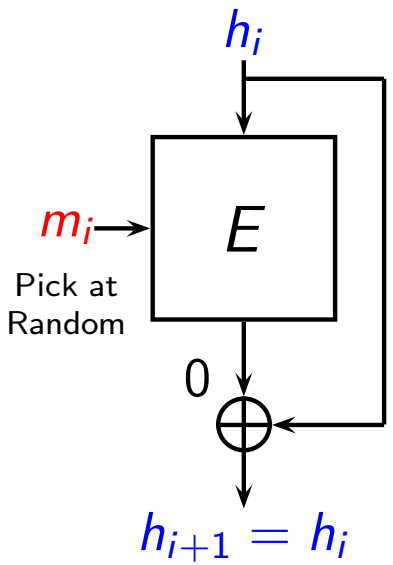
- Find $O(2^{m_c/2})$ fix-points denoted by $A = (h, m)$.
- Select $O(2^{m_c/2})$ single blocks and compute $B = (C_{MD}(IV, \tilde{m}), \tilde{m})$.
- Find a collision between A and B.
- Voilà — an **expandable message** $\tilde{m}||m^t$ for all t lead to the same chaining value h.



Generic Attacks against MD Construction (cont.)

- Fixpoint and second preimage attacks (long messages) — $2^{m_c - l}$, $O(2^{m_c/2})$ (Dean, 1999) [mainly for Davies-Meyer functions]

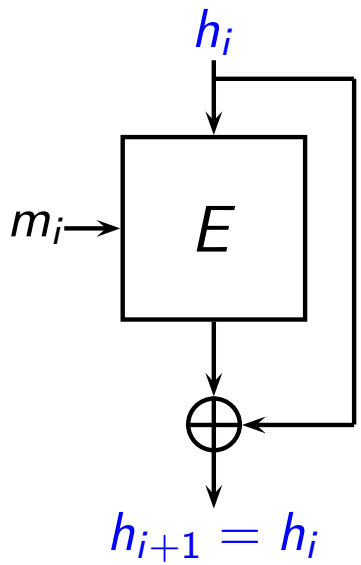
- Find $O(2^{m_c/2})$ fix-points denoted by $A = (h, m)$.
- Select $O(2^{m_c/2})$ single blocks and compute $B = (C_{MD}(IV, \tilde{m}), \tilde{m})$.
- Find a collision between A and B.
- Voilà — an **expandable message** $\tilde{m}||m^t$ for all t lead to the same chaining value h.



Generic Attacks against MD Construction (cont.)

- Fixpoint and second preimage attacks (long messages) — $2^{m_c - l}$, $O(2^{m_c/2})$ (Dean, 1999) [mainly for Davies-Meyer functions]

- Take the message M.
- Starting from h, try to find a message block x s.t., $f(h, x) = h_i$, for one of the chaining values of M.
- If succeeded, pad the message to the right length and obtain a second preimage.



Generic Attacks against MD Construction (cont.)

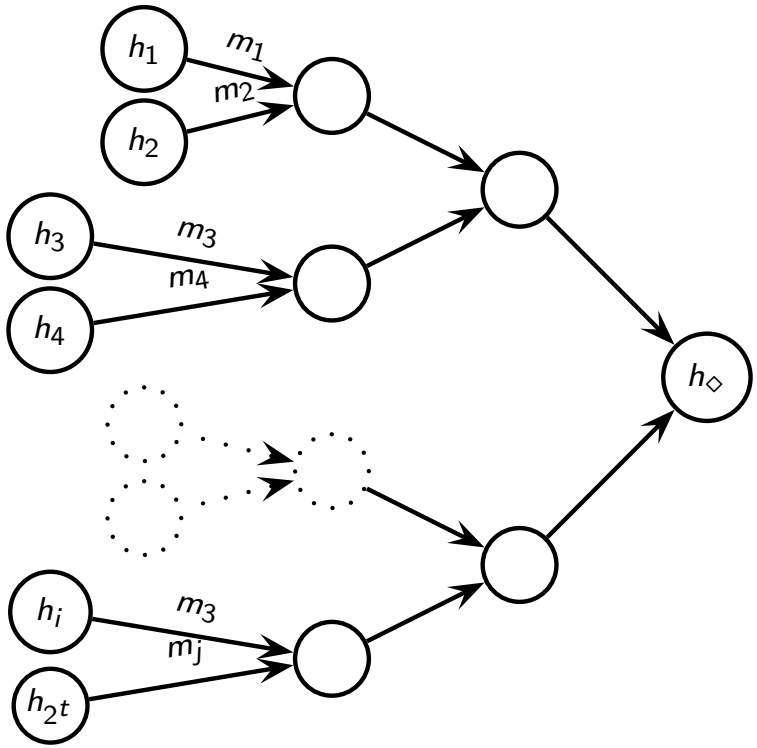
- One-of-many second preimage attacks — $2^{m_c - kl}$, $O(2^{m_c/2})$ (Dean, 1999)
- Expandable messages and second preimage attacks (long messages) — $2^{m_c - l}$, $O(2^{m_c/2})$ (Kelsey, Schneier, 2005) [any iterative hash function]. They forgot to mention, applicable also to one-of-many second preimage attacks — $2^{m_c - kl}$, $O(2^{m_c/2})$

- In [KS05] the expandable message is constructed as a multi-collision. In the first block between a message of one block and a message of two blocks, then between one block and three blocks, one and five, etc.

Generic Attacks against MD Construction (cont.)

- Targeted preimage attack (herding attack) — $O(2^{m_c - t} + 2^{(m_c+t)/2})$ (Kelsey, Kohno, 2006).

Precomputation — generation of a **diamond structure**.



Generic Attacks against MD Construction (cont.)

- Targeted preimage attack (herding attack) — $O(2^{m_c-t} + 2^{(m_c+t)/2})$ (Kelsey, Kohno, 2005).
- After committing to $h_\diamond$, the attacker is given a prefix P, and has to find a message M, such that $H_f(M) = h_\diamond$.
- The attacker tries 2^{m_c-t} possible x's until $H_f(P||x)$ is one of the precomputed h_i's.
- Then, by concatenating the path in the diamond structure to $P||x$ it is possible to find a preimage of H_f.

Generic Attacks against MD Construction (cont.)

- New long second preimage attack — $O(2^{m_c-t} + 2^{(m_c+t)/2} + 2^{m_c-l})$ (Andreeva et al., 2008).
- The attack of Kelsey and Schneier replaces about half of the message.
- A shorter "patch" can be done using diamond structures.
- Generate a diamond structure.
- Given M for which a second preimage has to be computed, try random m_{link2}, until $f(h_\diamond, m_{link2}) = h_i$, for some h_i obtained during the computation of $H_f(M)$.
- The patch is going to be $t+2$ blocks long (t is the length inside the diamond structure). So starting from h_{i-t-2}, try random m_{link1} until one of the entry points of the diamond structure are found.

Possible Conclusions

- We have to accept that second preimages are as easy to find as collisions.

Possible Conclusions

- We have to accept that second preimages are as easy to find as collisions.
- Maybe m_c should be further increased if we require that the security against second preimage attacks should be like against preimage attacks?

Possible Conclusions

- We have to accept that second preimages are as easy to find as collisions.
- Maybe m_c should be further increased if we require that the security against second preimage attacks should be like against preimage attacks?

Bounding the size of a message by at most $2^{m/2}$ blocks (or bits) — does not help, as the current limit is even smaller (2^{64} bits$= 2^{55}$ blocks). Also, most of the above mentioned attacks can deal with multiple messages without any additional cost.

Outline

The Construction of HAIFA

The Construction of HAIFA

The HAIFA Construction

- Major features:
  - Supports salts (defines families of hash functions).
  - Supports variable output size.
  - Offers as good security properties as can be.
  - Strong backward compatibility.
  - All suggested modes can be realized as HAIFA.

The HAIFA Compression Function

- Accepts as inputs:
  - A chaining value (of size m_c)
  - A message block (of size n)
  - A bit counter (of size b)
  - A salt (of size s)
- $f : \{0,1\}^{m_c} \times \{0,1\}^n \times \{0,1\}^b \times \{0,1\}^s \longrightarrow \{0,1\}^{m_c}$.

The HAIFA Initialization

- Let m be the target digest size.
- Let IV be a general initial value.
- $IV_m = C(IV, m, 0, 0)$.

The HAIFA Computation

- Take M, the message, and pad it:
  - Pad a single bit of 1.
  - Pad as many 0 bits as needed such that the length of the padded message (with the 1 bit and the 0's) is congruent modulo n to $(n - (t + r))$.
  - Pad the message length encoded in t bits.
  - **Pad the digest size encoded in r bits.**
- Set $h_0 = IV_m$
- For $i = 1, 2, \ldots, l$ compute $h_i = C(h_{i-1}, M_i, \#bits, salt)$.
- Truncate h_l to m bits.

Protection Against the Second Preimage Attacks

Possible solution proposed by Kelsey and Schneier is to add the block index as an input to the compression function, i.e.,

$$h_i = C(h_{i-1}, M_i, i),$$

instead of

$$h_i = C(h_{i-1}, M_i).$$

But the block index is never kept nowadays in the state of the hash.

Protection Against the Second Preimage Attacks

Possible solution proposed by Kelsey and Schneier is to add the block index as an input to the compression function, i.e.,

$$h_i = C(h_{i-1}, M_i, i),$$

instead of

$$h_i = C(h_{i-1}, M_i).$$

But the block index is never kept nowadays in the state of the hash.
The number of bits is kept.

Protection Against the Second Preimage Attacks

Possible solution proposed by Kelsey and Schneier is to add the block index as an input to the compression function, i.e.,

$$h_i = C(h_{i-1}, M_i, i),$$

instead of

$$h_i = C(h_{i-1}, M_i).$$

But the block index is never kept nowadays in the state of the hash.
The number of bits is kept.
This prevents one fix-point from being concatenated to itself.

Protection Against the Second Preimage Attacks (cont.)

- This solution protects against the second preimage attacks, as each computed block becomes **static** — cannot be used in any other location of any message, just at the original location.
- This completely foils the attack of Andreeva et al.
- It also ensures that there will not be cycles of chaining values (even if there is some value twice, e.g., as a fix-point, it will not be repeated).
- Notice that it is **not needed to increase** the chaining size m_c any more.

Protection Against Message Expansion

- The previous solution also protects against the ability to append data at end of a message, without knowing the message (e.g., when the hash function is used for MAC).
- This follows the fact that the last call to the compression function is either with a non-multiple of the block size, or with a zero.

[similar property would not hold if the block index would be used]

Variable Hash Size

Advantages

1. A general method for truncation of the hash size for any hash function
2. IV_m can be computed in advance, so that no extra time is required to hash a message (just like it is done in SHA-384, SHA-512)
3. On the other hand, applications that need several hash sizes, can compute IV_m on the fly, or actually compute

$$\text{truncate}_m(\text{HashFromIV}(m\|M)).$$

Variable Hash Size (cont.)

A potential problem

- The compression function is the same for all these functions.
- An attacker may wish to have the same hash value for two different hash sizes (up to truncation).
- If the attacker has the freedom to select different prefixes, he may apply a collision attack (or birthday attack) on the first few blocks, getting the same chaining value. The rest of the chaining values will be the same, as well as the hash value (up to truncation).
- In order to solve this possible problem, we add the digest size to the padding, which forces the attacker to attack the last compression function call to find such instances.

Family of Hash Functions and Salts

- To further reduce the susceptibility of the hash function to second preimage attack, we use salts.
- Each message has a different salt (chosen by the user).

Family of Hash Functions and Salts

- To further reduce the susceptibility of the hash function to second preimage attack, we use salts.
- Each message has a different salt (chosen by the user).
- This also satisfies that theoretic definitions of hash functions (**families** of hash functions $h_i()$).

Family of Hash Functions and Salts

- To further reduce the susceptibility of the hash function to second preimage attack, we use salts.
- Each message has a different salt (chosen by the user).
- This also satisfies that theoretic definitions of hash functions (**families** of hash functions $h_i()$).
- Such a modified definition of a cryptographic hash function should protect against one-of-many preimages, and other attacks that use several messages generated by the legal user.

Family of Hash Functions and Salts (cont.)

- The salt should be used as an additional input to the compression function (but not change the IV), and should be added to the padding (i.e., the last block is processed using the salt).
- Some applications that cannot select a hash function from a family, due to application dependent requirements, may select a fixed member (e.g., salt=0).
- The cost of this solution is negligible.
- In some sense, salt can be viewed as a nonce.
- Salt can also be used as a key for keyed hash functions — may be possible to design unified HASH/MAC designs.

Outline

Implementing Other Suggested Modes in HAIFA

- The optimality of HAIFA was not acknowledged by all, and we have **competition**!
- But that is OK, because all suggested modes can be implemented as an instantiation of HAIFA.
- The only thing you may lose by choosing to implement HAIFA in a different mode is . . . security.

Randomized Hashing

- Halevi & Krawczyk suggested the following randomized hashing:

$$H_r(M_1||M_2||\ldots||M_k) \stackrel{def}{=} H(M_1 \oplus r||M_2 \oplus r||\ldots||M_k \oplus r)$$
$$\tilde{H}_r(M) \stackrel{def}{=} H_r(0||M) = H(r||M_1 \oplus r||M_2 \oplus r||\ldots||M_k \oplus r)$$

- In HAIFA:

$$C_{HAIFA1}(h_{i-1}, M_i, \#bits, s) = C_{MD}(h_{i-1}, M_i \oplus s),$$
$$C_{HAIFA2}(h_{i-1}, M_i, \#bits, s) = \begin{cases} C_{MD}(C_{MD}(h_{i-1}, s), M_i \oplus s) & \text{First Block} \\ C_{MD}(h_{i-1}, M_i \oplus s) & \text{Otherwise} \end{cases}$$

- Security loss: second preimage attacks (long, herding), herding, fixpoints.

Enveloped Merkle-Damgård

- Suggested by Bellare & Ristenpart:
  - Changed padding scheme. 1 and as many 0's as needed are appended such that the length is equal to $n - m_c \bmod n$.
  - The final digest value is computed as

$$h_k = C_{MD}(IV_2, h_{k-1}||M_k),$$

    for a second initialization vector IV_2.
- In HAIFA:

$$C_{HAIFA3}(h_{i-1}, M_i, \#bits, s) = \begin{cases} C_{MD}(IV_2, h_{i-1}||fix\_pad(M_i)) & \text{Last Block} \\ C_{MD}(h_{i-1}, M_i) & \text{Otherwise} \end{cases}$$

- Security loss: second preimage attacks (long, herding), herding, fixpoints.

ROX – Property Preserving Transformation

- Suggested by Andreeva et al.
  - Pad M using $RO_2(K, s, M)$, where K is the key (of length k bits) and s is the salt.
  - Set $h_0 = IV$.
  - Generate $\mu_i = RO_1(K, s, i)$, for $i = 0, \ldots, \lceil \log_2(|M|) \rceil$.
  - For $i = 1, \ldots, l$ compute $h_i = C_{RMC}(h_{i-1} \oplus \mu_{\nu(i)}, M_i, K)$.
- In HAIFA

$$C_{HAIFA4}(h_{i-1}, M_i, \#bits, s) = \begin{cases} C_{RMC}(h_{i-1} \oplus RO_1(s_1, s_2, \nu(i)), M_i || pad_{RO2(s_1,s_2)}) & \text{Last block} \\ C_{RMC}(h_{i-1} \oplus RO_1(s_1, s_2, \nu(i)), M_i, s_1) & \text{Otherwise} \end{cases}$$

- Security loss: Herding, long second preimage based on herding (both logarithmic gain over MD).

Dithered Hash

- Suggested by Rivest.
  - Each compression function call is invoke with a dither character
$$h_{i+1} = C_{MD}(h_i, m_i, d(i))$$
  - For all but the last block $d(i) = 0||K[\lfloor i/2^{13} \rfloor]||i \bmod 2^{13}$.
  - For the last block $d(i) = 1||K[\lfloor i/2^{13} \rfloor]||i \bmod 2^{13}$.
- Well, you can use the same compression function in HAIFA.
- Security loss: Herding, long second preimage based on herding (still there is a gain of depending on the dither sequence).

New Results on the Security of HAIFA

- Collision resistance — if the compression function is collision resistant then so does the hash function.

New Results on the Security of HAIFA

- Collision resistance — if the compression function is collision resistant then so does the hash function.
- According to [APNS07], HAIFA does not preserve the Sec and the Pre properties of the compression function.

New Results on the Security of HAIFA

- Collision resistance — if the compression function is collision resistant then so does the hash function.
- According to [APNS07], HAIFA does not preserve the Sec and the Pre properties of the compression function.
- However, we show that if the compression function is really good, i.e., a random oracle or an ideal cipher — we can prove that the second preimage resistance of HAIFA is optimal (i.e., $O(2^n)$ work is needed to find a second preimage).

Second Preimage Resistance of HAIFA

- If the compression function is ideal, all the chaining values are random.
- If the second preimage is of a different length, then the attacker found a second preimage of the last compression function (which takes 2^n).
- Otherwise, one of the calls to the compression function hit one internal chaining value.
- As the chaining values are all random, and as each block has a "different" function, this causes the attacker to succeed only by chance.

Can We Weaken the Assumption?

- Any weaker assumption on the compression function does not scale too well.
- We show that the second preimage resistance drops with the length of the message (specifically $2^n/l$).
- This is due to the best "game":

Can We Weaken the Assumption?

- Any weaker assumption on the compression function does not scale too well.
- We show that the second preimage resistance drops with the length of the message (specifically $2^n/l$).
- This is due to the best "game":
  - Let A be a second preimage adversary for the hash function.
  - A commits to some message she is going to find a second preimage.
  - Then, B, out algorithm guesses which of the blocks A is going to generate a second preimage to, and commits to this block.
  - B is given the key, and passes it on to A.
  - If A indeed found a second preimage in the expected position — B succeeded.

Can We Weaken the Assumption? (cont.)

- Thus, all the modes, the longer the message, the second preimage resistance drops.
- But this loss is only in the sense that we do not know how to prove that it is secure!

Outline

Summary

- HAIFA proposes a secure framework for designing hash functions.
- If your compression function works in HAIFA, you can later adapt it to any other iterative mode.
- HAIFA offers protection against a wide range of attacks, and along with the salts, can be used to instantiate any needed knife — simple, or swiss.

Summary — Second-Preimage Resistance

Mode	Provable Lower Bound	Best Known Attack
Merkle-Damgård	$2^n/l$	$2^n/l$
Randomized-Hashing	$2^n/l$	$2^n/l$
Shoup/ROX	$2^n/l$	$\log(n) \cdot 2^n/l$
Dithering	$2^n/l$	$2^n \cdot \lvert\mathcal{D}^*\rvert/l$
HAIFA	$2^n/l$	2^n

Open Issues

- Closing the gap between the proof and the actual security.
- Find good definitions concerning the choice of the salt.
- Security without salts — definitions and proofs.

Questions?

Thank you for your attention!