Herding Hash Functions and the Nostradamus Attack

Presented by Ohad Lutzky, University of Haifa

John Kelsey[1] Tadayoshi Kohno[2]

[1]National Institue of Standards and Technology

[2]CSE Department, UC San Diego

Cryptanalysis of Hash Functions Seminar, Spring 2011

Outline

Outline

Using hash functions for commitments

- Commit to knowledge of a message M
- Do not reveal the message (for now)
- Solution: Reveal Hash(M)
- Safety: Preimage resistance

Example: Uri Geller

Uri Geller Say:

I, Uri Geller, have predicted many important predictions about the distant future, as well as a closer event: The closing prices of all stocks traded in the Tel Aviv Stock Exchange on the last day of 2012.

- Uri could place his predictions in an envelope in a safe.
- Instead, he provides the MD5 hash H of the entire prediction.

Wait, isn't MD5 broken?

- Collision-resistance for MD5 has been compromised
- However, we only need *preimage* resistance for this scheme
- ...or do we?

Wait, isn't MD5 broken?

- Collision-resistance for MD5 has been compromised
- However, we only need *preimage* resistance for this scheme
- ...or do we?

Outline

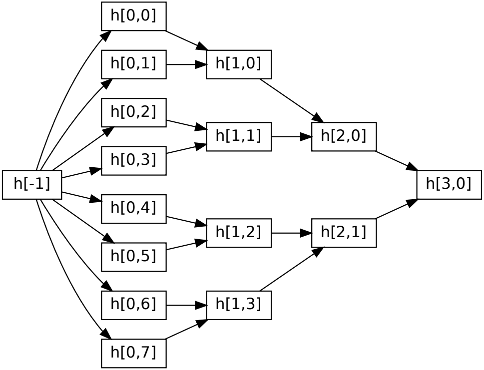

Figure: The diamond structure

- Applicable to Merkle-Damgård-style hashes
- Edges represent message blocks
- Vertices ($h[i,j]$) represent hash values
- Width of stages: $2^k, 2^{k-1}, \ldots, 2, 1$.

Outline

Producing a suffix from an intermediate hash value

- Suppose you've somehow reached the intermediate hash value $h[0,2]$.
- Append the blocks represented by the edges:
  - $h[0,2] \rightarrow h[1,1]$
  - $h[1,1] \rightarrow h[2,0]$
  - $h[2,0] \rightarrow h[3,0]$
- Your new final hash value is $h[3,0]$.

Outline

Cost of construction (cheaper than you'd expect)

- Mapping 2^k hash values down to 2^{k-1}:
  - Generate about $2^{n/2+1/2-k/2}$ candidates.
  - Look for collisions.
- Total work is about $2^{n/2+k/2+2}$.
- Parallelizable using technique by P. van Oorschot and M. Wiener.

Employing cryptanalytic attacks

- Use weaknesses in collision-resistance of the compression function
- An algorithm which only works for an identical IV doesn't help
- An algorithm which works for any known IV difference works best
- An algorithm which works for a subset of IV pairs is still useful
  - ...if those pairs can be recognized efficiently.

Precomputation of the prefix

- The set of possible prefixes may be known and small
- If so, build the diamond directly from their intermediate hashes.

Outline

Attack plan

1. *Build the diamond structure*. Commit to $H = h[k, 0]$.
   - You have plenty of time to do this.
2. *Determine the prefix P:* Wait for the event you were predicting the results of.
3. *Find a linking message:* Search for a single-block message M to append to P, s.t. the intermediate hash of $P||M$ is in the search structure.
4. *Produce the message:* Use the search struction to find a suffix S s.t. hash$(P||M||S) = H$.

Attack plan

1. *Build the diamond structure*. Commit to $H = h[k, 0]$.
   - You have plenty of time to do this.
2. *Determine the prefix P:* Wait for the event you were predicting the results of.
3. *Find a linking message:* Search for a single-block message M to append to P, s.t. the intermediate hash of $P||M$ is in the search structure.
4. *Produce the message:* Use the search struction to find a suffix S s.t. hash$(P||M||S) = H$.

Attack plan

1. *Build the diamond structure*. Commit to $H = h[k, 0]$.
   - You have plenty of time to do this.
2. *Determine the prefix P:* Wait for the event you were predicting the results of.
3. *Find a linking message:* Search for a single-block message M to append to P, s.t. the intermediate hash of $P||M$ is in the search structure.
4. *Produce the message:* Use the search struction to find a suffix S s.t. hash$(P||M||S) = H$.

Attack plan

1. *Build the diamond structure*. Commit to $H = h[k, 0]$.
   - You have plenty of time to do this.
2. *Determine the prefix P:* Wait for the event you were predicting the results of.
3. *Find a linking message:* Search for a single-block message M to append to P, s.t. the intermediate hash of $P||M$ is in the search structure.
4. *Produce the message:* Use the search struction to find a suffix S s.t. hash$(P||M||S) = H$.

Attack plan

1. *Build the diamond structure*. Commit to $H = h[k, 0]$.
   - You have plenty of time to do this.
2. *Determine the prefix P:* Wait for the event you were predicting the results of.
3. *Find a linking message:* Search for a single-block message M to append to P, s.t. the intermediate hash of $P||M$ is in the search structure.
4. *Produce the message:* Use the search struction to find a suffix S s.t. hash$(P||M||S) = H$.

Finding a linking message

- We want to produce a final hash of H
- Our diamond structure can get us from any hash value in $h[0, ?]$ to H.
- Therefore, we need a linking block M, so that the intermediate hash of $P||M$ is in $h[0, ?]$.
- Expected tries: 2^{n-k}.
- Not necessary if we've created the diamond structure from a known set of prefixes.

Outline

A note on length

- Merkle-Damgård strengthening means message length is taken into account. (It's appended to the last block)
- Messages are appended blockwise
- Therefore, while not explicitly stated in the article, either:
  - P is assumed to be a fixed length, or padded to such a length.
  - $|P|$ will have to be an integer number of blocks, $n \cdot |B|$.
- The length of our final message will be:

$$|P| + \underbrace{1 \cdot |B|}_{\text{Linking message}} + \underbrace{(k+1) \cdot |B|}_{|S|}$$

  - ...$-1 \cdot |B|$ if we don't need a linking block.
  - ...and if so, we can't use it at all.

Expandable messages

- J. Kelsey and B. Schneier discuss (a, b)-expandable messages
- This is a set of messages between lengths a and b with the same intermediate hash
- Can be efficiently found for MD5, SHA1 and others, for about twice the cost of brute-force
- To use *all* intermediate hash values $h[?, ?]$ in the structure, a $(1, k+1)$-expandable message must be produced at its end.
- Otherwise, only the widest layer (2^k) can be used
- This is a note of discrepancy, further analyzed by Ross and Shrimpton.

Total work done

- Generating the diamond structure: $2^{n/2+k/2+2}$
  - Or less, with cryptographic attacks
- Finding the linking message: 2^{n-k}.
- For the example results (using expandable messages):
  - Work is $2^{n-k-1} + 2^{n/2+k/2+2} + k \times 2^{n/2+1}$
  - $k = \frac{n-5}{3}$ is found to be ideal, giving $W \approx 2^{n-k}$.

Results (theoretical)

Output Size	Function	k	Suffix blocks	Work
n		$(n-5)/3$	$k + \lg(k) + 1$	2^{n-k}
128	MD5	41	48	2^{87}
160	SHA1	52	59	2^{108}
192	Tiger	63	70	2^{129}
256	SHA256	84	92	2^{172}
512	Whirlpool	169	178	2^{343}

Making messages meaningful

- Use Gideon Yuval's trick (*How to swindle Rabin*, 1979)
- A block with many variation points can be used to generate many equivalent-meaning messages for each diamond layer
- Suffixes will be (much) longer, but not harder to find
- This is made easier because we commit to *meaning*, not bits.

This prophecy/information has been/was brought forth/to me by the heavens/angels...

Outline

Commit to an ordering (Hash Router)

- Prove (perhaps when gambling) to be able to predict the outcome of a 32-entrant race.
- Commit to 32 hash outputs $H_0, H_1, \ldots, H_{31}$.
- After the race is over, produce output strings $S_0, \ldots, S_{31}$
  - S_i describes the entrant in the race who finished ith
  - $H_i = \text{hash}(S_i)$
- Perform like so:
  - Create a diamond structure herding to H
  - When creating the diamond, start with entrant names
  - Append strings “finishes 1st”, “finishes 2nd”, ..., and commit to the resulting hashes
  - When learning results, herd to H and append appropriate string

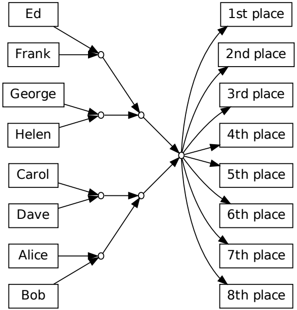

Figure: A “Hash Router”

Outline

Predicting the future the Uri Geller/Nostradamus attack

- Claim psychic power / future telling
- Claim higher understanding of science / economics
- “Prove” access to insider information

Steal credit for inventions

- Periodically submit a hash to a digital timestamping service
- Learn of an amazing invention
- Create a message describing the invention, and make sure it hashes to a hash submitted in the past
- To save computation, create the diamond once, and simply add one block to the end every submission

Create tweakable signatures

- Create a document to sign using a diamond structure
- The “prefix” portion can later be modified, without changing the hash value.

Summary

- Collision resistance is more important than you might think.
- Do not trust messages with wonky suffixes.
- When receiving a commitment, specify a rigid format.

Created with

- LaTeX-BEAMER
  - Using `mathserif`, math font is `eulervm`
- Graphviz
- Vim