Cryptographic Hash Functions
Part II

Cryptography 1

Andreas Hülsing, TU/e
Some slides by Sebastiaan de Hoogh, TU/e
Hash function design

- Create fixed input size building block
- Use building block to build compression function
- Use „mode“ for length extension

Engineering
- Permutation / Block cipher
- Cryptanalysis / best practices

Generic transforms
- Compression function
- Hash function
- Reductionist proofs
(LENGTH-EXTENSION) MODES
Merkle-Damgård construction

Given:
• compression function: \( CF : \{0,1\}^n \times \{0,1\}^r \rightarrow \{0,1\}^n \)

Goal:
• Hash function: \( H : \{0,1\}^* \rightarrow \{0,1\}^n \)
Merkle-Damgård - iterated compression
Merkle-Damgård construction

- assume that message $m$ can be split up into blocks $m_1, \ldots, m_s$ of equal block length $r$
  - most popular block length is $r = 512$
- compression function: $CF : \{0,1\}^n \times \{0,1\}^r \rightarrow \{0,1\}^n$
- intermediate hash values (length $n$) as $CF$ input and output
- message blocks as second input of $CF$
- start with fixed initial $IHV_0$ (a.k.a. $IV = initialization \ vector$)
- iterate $CF : IHV_1 = CF(IHV_0, m_1), IHV_2 = CF(IHV_1, m_2), \ldots, IHV_s = CF(IHV_{s-1}, m_s)$,
- take $h(m) = IHV_s$ as hash value
- advantages:
  - this design makes streaming possible
  - hash function analysis becomes compression function analysis
  - analysis easier because domain of $CF$ is finite
padding

• *padding*: add dummy bits to satisfy block length requirement

• non-ambiguous padding: add one 1-bit and as many 0-bits as necessary to fill the final block
  – when original message length is a multiple of the block length, apply padding anyway, adding an extra dummy block
  – any other non-ambiguous padding will work as well
Merkle-Damgård strengthening

- let padding leave final 64 bits open
- encode in those 64 bits the original message length
  - that’s why messages of length $\geq 2^{64}$ are not supported
- reasons:
  - needed in the proof of the Merkle-Damgård theorem
  - prevents some attacks such as
    - trivial collisions for random $IV$
      - now $h(IHV_0, m_1 || m_2) = h(IHV_1, m_2)$
    - see next slide for more
Merkle-Damgård strengthening, cont’d

- fixpoint attack
  
  fixpoint: \( IHV, m \) such that \( CF(IHV,m) = IHV \)

- long message attack
**compression function collisions**

- **collision** for a compression function: \( m_1, m_2, IHV \) such that 
  \[
  CF(IHV,m_1) = CF(IHV,m_2)
  \]

- **pseudo-collision** for a compression function: \( m_1, m_2, IHV_1, IHV_2 \) such that 
  \[
  CF(IHV_1,m_1) = CF(IHV_2,m_2)
  \]

- **Theorem** (Merkle-Damgård): If the compression function \( CF \) is pseudo-collision resistant, then a hash function \( h \) derived by Merkle-Damgård iterated compression is collision resistant.
  - Proof: Suppose \( h(m_1) = h(m_2) \), then
    - If \( m_1 \) and \( m_2 \) same size: locate the iteration where the collision occurs
    - Else a pseudo collision for \( CF \) appears in the last blocks (cont. length)

- **Note:**
  - a method to find pseudo-collisions does not lead to a method to find collisions for the hash function
  - a method to find collisions for the compression function is almost a method to find collisions for the hash function, we ‘only’ have a wrong \( IHV \)
Sponges

Given:
• permutation: \( f : \{0,1\}^b \rightarrow \{0,1\}^b \)

Goal:
• Hash function: \( H : \{0,1\}^* \rightarrow \{0,1\}^n \)
  (actually \( H : \{0,1\}^* \rightarrow \{0,1\}^* \))

• (Already includes CF design, more later)
Sponges

- Used and introduced in SHA3 aka Keccak
  - Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche
Intercourse: Random oracles

• Models the perfect hash function
• Truely random function without any structure
• Best attacks: Generic attacks (No structure available!)

Issue:
• No way to build a RO with polynomial description

Mind Model:
• Lazy-sampling
  – Imagine a black box implementing the function
  – For every new query, a random response is sampled
  – For old queries, former response is used
Sponge security

- **Theorem (Indifferentiability from a random oracle):**
  If \( f \) is a random permutation, the expected complexity for differentiating a sponge from a random oracle is \( \sqrt{\pi} \frac{2^c}{2} \).

- **Note:**
  - Neat way to simplify security arguments
  - Implies bounds for all attacks that use less than \( \sqrt{\pi} \frac{2^c}{2} \) queries
  - Bounds are those of generic attacks against a random oracle
COMPRESSION FUNCTION
DESIGN
Block-Cipher-based designs

- Traditional approach
- Many possible modes
  - see Preneel, Govaerts, Vandewalle. Hash functions based on block ciphers: a synthetic approach. CRYPTO’93
  - security: Black, Rogaway, Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. CRYPTO’02
- Most popular: Matyas-Meyer-Oseas
Permutation-based designs

- Less frequent use
- Keccak compression function:

\[ M_i \]

\[ \text{IHV}_i \rightarrow \text{IHV}_{i+1} \]

- Important: NEVER hand out bits last c bits of IHV!
Security

• Generally analyzed in idealized models:
  – „Black-box models“
  – Ideal cipher model
  – Random oracle model
  – Random permutation model

• Proofs assuming underlying building block behaves like such an idealized building block
BASIC BUILDING BLOCKS
the MD4 family of hash functions

- **MD4** (Rivest 1990)
  - **RIPEMD** (RIPE 1992)
    - **RIPEMD-128**
    - **RIPEMD-160**
    - **RIPEMD-256**
    - **RIPEMD-320** (Dobbertin, Bosselaers, Preneel 1992)
  - **MD5** (Rivest 1992)
  - **HAVAL** (Zheng, Pieprzyk, Seberry 1993)
    - **SHA-0** (NIST 1993)
    - **SHA-1** (NIST 1995)
      - **SHA-224**
      - **SHA-256**
      - **SHA-384**
      - **SHA-512** (NIST 2004)
design of MD4 family compression functions

message block
split into words
message expansion
input words for each step

$IHV \rightarrow$ initial state
each step updates state with an input word
final state ‘added’ to $IHV$

(feed-forward)
design details

• MD4, MD5, SHA-0, SHA-1 details:
  – 512-bit message block split into 16 32-bit words
  – state consists of 4 (MD4, MD5) or 5 (SHA-0, SHA-1) 32-bit words
  – MD4: 3 rounds of 16 steps each, so 48 steps, 48 input words
  – MD5: 4 rounds of 16 steps each, so 64 steps, 64 input words
  – SHA-0, SHA-1: 4 rounds of 20 steps each, so 80 steps, 80 input words
  – message expansion and step operations use only very easy to implement operations:
    • bitwise Boolean operations
    • bit shifts and bit rotations
    • addition modulo $2^{32}$
  – proper mixing believed to be cryptographically strong
message expansion

- **MD4, MD5** use *roundwise permutation*, for MD5:
  - \( W_0 = M_0, \ W_1 = M_1, \ldots, \ W_{15} = M_{15}, \)
  - \( W_{16} = M_1, \ W_{17} = M_6, \ldots, \ W_{31} = M_{12}, \) (jump 5 mod 16)
  - \( W_{32} = M_5, \ W_{33} = M_8, \ldots, \ W_{47} = M_2, \) (jump 3 mod 16)
  - \( W_{48} = M_0, \ W_{49} = M_7, \ldots, \ W_{63} = M_9 \) (jump 7 mod 16)

- **SHA-0, SHA-1** use *recursivity*
  - \( W_0 = M_0, \ W_1 = M_1, \ldots, \ W_{15} = M_{15}, \)
  - SHA-0: \( W_i = W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16} \) for \( i = 16, \ldots, 79 \)
  - problem: \( k^{\text{th}} \) bit influenced only by \( k^{\text{th}} \) bits of preceding words, so not much diffusion
  - SHA-1: \( W_i = (W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W_{i-16}) \ll 1 \)
    (additional rotation by 1 bit,
    this is the *only* difference between SHA-0 and SHA-1)
Example: step operations in MD5

- in each step only one state word is updated
- the other state words are *rotated* by 1
- state update:

\[
A' = B + ((A + f_i(B,C,D) + W_i + K_i) \ll s_i)
\]

- \(K_i, s_i\) step dependent constants,
- + is addition mod \(2^{32}\),
- \(f_i\) round dependent boolean functions:

\[
f_i(x,y,z) = xy \ OR (\neg x)z \text{ for } i = 1, ..., 16,
\]

\[
f_i(x,y,z) = xz \ OR y(\neg z) \text{ for } i = 17, ..., 32,
\]

\[
f_i(x,y,z) = x \ XOR y \ XOR z \text{ for } i = 33, ..., 48,
\]

\[
f_i(x,y,z) = y \ XOR (y \ OR (\neg z)) \text{ for } i = 49, ..., 64,
\]

- these functions are nonlinear, balanced, and have an *avalanche effect*
step operations in MD5
provable hash functions

- people don’t like that one can’t prove much about hash functions
- reduction to established ‘hard problem’ such as factoring is seen as an advantage
- Example: VSH – Very Smooth Hash
  - Contini-Lenstra-Steinfeld 2006
  - collision resistance provable under assumption that a problem directly related to factoring is hard
  - but still far from ideal
    - bad performance compared to SHA-256
    - all kinds of multiplicative relations between hash values exist
    - not post-quantum secure
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Snefru</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIPEMD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HAVAL-128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA-0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIPEMD-128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[1] RIPEMD-128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIPEMD-160</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA-2 family</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA-3 (Keccak)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Key:** Unbroken, Weakened, Broken, Deprecated

[1] Note that 128-bit hashes are at best $2^{64}$ complexity to break; using a 128-bit hash is irresponsible based on sheer digest length.

[2] In 2007, the [NIST launched the SHA-3 competition](http://www.nist.gov/sha3) because "Although there is no specific reason to believe that a practical attack on any of the SHA-2 family of hash functions is imminent, a successful collision attack on an algorithm in the SHA-2 family could have catastrophic effects for digital signatures." One year later the first strength reduction was published.

[The Hash Function Lounge](http://www.hashfunctionlounge.com) has an excellent list of references for most of the dates. Wikipedia now has references to the rest.
Real life attacks on MD5
Example Hash-then-Sign in Browser
Wang’s attack on MD5

• **two-block** collision
  – for any input $IHV$, identical for the two messages
    i.e. $IHV_0 = IHV_0', \Delta IHV_0 = 0$
  – **near-collision** after first block:
    $IHV_1 = CF(IHV_0, m_1), IHV_1' = CF(IHV_0, m_1'),$
    with $\Delta IHV_1$ having only a few carefully chosen $\pm1$s
  – full collision after second block:
    $IHV_2 = CF(IHV_1, m_2) = CF(IHV_1', m_2'),$
    i.e. $IHV_2 = IHV_2', \Delta IHV_2 = 0$

• with $IHV_0$ the standard IV for MD5, and a third block
  for padding and MD-strengthening, this gives a collision for the full MD5
chosen-prefix collisions

- latest development on MD5
- Marc Stevens (TU/e MSc student) 2006
  - paper by Marc Stevens, Arjen Lenstra and Benne de Weger, EuroCrypt 2007
- Marc Stevens (CWI PhD student) 2009
  - paper by Marc Stevens, Alex Sotirov, Jacob Appelbaum, David Molnar, Dag Arne Osvik, Arjen Lenstra and Benne de Weger, Crypto 2007
  - rogue CA attack
MD5: identical IV attacks

- all attacks following Wang’s method, up to recently
- MD5 collision attacks work for any starting \textit{IHV}
  data before and after the collision can be \textit{chosen at will}
- but starting \textit{IHVs} must be identical
  data before and after the collision \textit{must be identical}
- called \textit{random collision}
MD5: different IV attacks

- **new attack**
  - Marc Stevens, TU/e
  - Oct. 2006
- **MD5 collisions for any starting pair** \(\{IHV_1, IHV_2\}\)
  - Data before the collision needs not to be identical
  - Data before the collision can still be chosen at will, for each of the two documents
  - Data after the collision still must be identical
- **called** chosen-prefix collision
indeed that was not the end
in 2008 the ethical hackers came by
observation: commercial certification authorities still use MD5

idea: proof of concept of realistic attack as wake up call
→ attack a real, commercial certification authority

purchase a web certificate for a valid web domain
  but with a “little tweak” built in
prepare a rogue CA certificate with identical MD5 hash
the commercial CA’s signature also holds for the rogue CA certificate
Outline of the RogueCA Attack
<table>
<thead>
<tr>
<th>Subject = End Entity</th>
<th>rogue CA certificate</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial number</td>
<td>serial number</td>
</tr>
<tr>
<td>commercial CA name</td>
<td>commercial CA name</td>
</tr>
<tr>
<td>validity period</td>
<td>validity period</td>
</tr>
<tr>
<td>domain name</td>
<td>rogue CA name</td>
</tr>
<tr>
<td>2048 bit RSA public key</td>
<td>1024 bit RSA public key</td>
</tr>
<tr>
<td>v3 extensions</td>
<td>v3 extensions</td>
</tr>
<tr>
<td>Subject = CA</td>
<td>Subject = CA</td>
</tr>
</tbody>
</table>

- chosen prefixes
- collision bits
- identical suffixes
- signature
- signature
problems to be solved

predict the serial number
predict the time interval of validity
   at the same time
   a few days before
more complicated certificate structure
   “Subject Type” after the public key
small space for the collision blocks
   is possible but much more computations needed
not much time to do computations
   to keep probability of prediction success reasonable
how difficult is predicting?

time interval:
CA uses automated certification procedure
certificate issued exactly 6 seconds after click

serial number:
Nov  3 07:44:08 2008 GMT   643006
Nov  3 07:45:02 2008 GMT   643007
Nov  3 07:46:02 2008 GMT   643008
Nov  3 07:47:03 2008 GMT   643009
Nov  3 07:48:02 2008 GMT   643010
Nov  3 07:49:02 2008 GMT   643011
Nov  3 07:50:02 2008 GMT   643012
Nov  3 07:51:12 2008 GMT   643013
Nov  3 07:51:29 2008 GMT   643014
Nov  3 07:52:02 2008 GMT   have a guess…
the attack at work

estimated: 800-1000 certificates issued in a weekend

procedure:

1. buy certificate on Friday, serial number S-1000
2. predict serial number S for time T Sunday evening
3. make collision for serial number S and time T: 2 days time
4. short before T buy additional certificates until S-1
5. buy certificate on time T-6

hope that nobody comes in between and steals our serial number S
to let it work

cluster of >200 PlayStation3 game consoles (1 PS3 = 40 PC’s)

complexity: $2^{50}$
memory: 30 GB

→ collision in 1 day
result

success after 4th attempt (4th weekend)

purchased a few hundred certificates
(promotion action: 20 for one price)
total cost: < US$ 1000
conclusion on MD5

- at this moment, ‘meaningful’ hash collisions are
  - easy to make
  - but also easy to detect
  - still hard to abuse realistically
- with chosen-prefix collisions we come close to realistic attacks
- to do real harm, second pre-image attack needed
  - real harm is e.g. forging digital signatures
  - this is not possible yet, not even with MD5
- More information: http://www.win.tue.nl/hashclash/
Questions?
proof of birthday paradox

- probability that all k elements are distinct is

\[
\prod_{i=0}^{k-1} \frac{t-i}{t} = \prod_{i=0}^{k-1} \left(1 - \frac{i}{t}\right) \leq \prod_{i=0}^{k-1} e^{-\frac{i}{t}} = e^{-\frac{k-1}{t}} e^{-\sum_{i=0}^{k-1} \frac{i}{t}} = e^{-\frac{k(k-1)}{2t}}
\]

and this is < ½ when \(k(k-1) > (2 \log 2)t\)

\(\approx k^2\) \(\approx 1.4 \times t\)