The Hash Function Hamsi

Özgül Küçük

Katholieke Universiteit Leuven

Second SHA-3 Conference
August 24, 2010
Outline

General Design Approach

Security of Hamsi

Software/Hardware Performance

Conclusion
General Design

General Design Approach
Security of Hamsi
Software/Hardware Performance
Conclusion

Ozgül Küçük
Katholieke Universiteit Leuven

The Hash Function Hamsi
Design Choices (1/3)
Design Choices (1/3)

- Inspired by stream based hash algorithms
Design Choices (1/3)

- Inspired by stream based hash algorithms
  - Short message blocks are processed with a light compression function in each iteration.
Design Choices (1/3)

- Inspired by **stream based** hash algorithms
  - Short message blocks are processed with a **light compression** function in each iteration.
  - Mixing of message blocks into the state takes **several iterations**.
Design Choices (1/3)

- Inspired by *stream based* hash algorithms
  - Short message blocks are processed with a *light compression* function in each iteration.
  - Mixing of message blocks into the state takes *several iterations*.
  - Security of compression function should be considered in the context of the iteration mode.
Design Choices (1/3)

- Inspired by **stream based** hash algorithms
  - **Short message** blocks are processed with a **light compression** function in each iteration.
  - **Mixing of message blocks** into the state takes **several iterations**.
  - **Security of compression function** should be considered in the context of the iteration mode.
- **Narrow-pipe design**
  - **Chaining value** has the same size as the digest length.
Design Choices (1/3)

- **Inspired by stream based hash algorithms**
  - Short message blocks are processed with a light compression function in each iteration.
  - Mixing of message blocks into the state takes several iterations.
  - Security of compression function should be considered in the context of the iteration mode.

- **Narrow-pipe design**
  - Chaining value has the same size as the digest length.
  - Hamsi-256/512 is mainly intended for users who want 128/256-bit security.
Design Choices (2/3)
Design Choices (2/3)

- Strong linear message expansion
Design Choices (2/3)

- Strong linear message expansion
  - Best Known Linear Codes (high minimum distance).
    - \([128, 16, 70]\) → Hamsi-256
    - \([256, 32, 131]\) → Hamsi-512
Design Choices (2/3)

- **Strong linear message expansion**
  - **Best Known Linear Codes** (high minimum distance).
    - $[128, 16, 70] \rightarrow \text{Hamsi-256}$
    - $[256, 32, 131] \rightarrow \text{Hamsi-512}$
  - **Flexible**
Design Choices (2/3)

- Strong linear message expansion
  - Best Known Linear Codes (high minimum distance).
    - [128, 16, 70] → Hamsi-256
    - [256, 32, 131] → Hamsi-512
  - Flexible
    - Can be implemented with a table of 1Kb
    - Or with a table of 32Kb (fast software)
    - Or by exploiting the structure of the code (compact hardware)
Design Choices (2/3)

- Strong linear message expansion
  - Best Known Linear Codes (high minimum distance).
    - $[128, 16, 70] \rightarrow \text{Hamsi-256}$
    - $[256, 32, 131] \rightarrow \text{Hamsi-512}$
  - Flexible
    - Can be implemented with a table of 1Kb
    - Or with a table of 32Kb (fast software)
    - Or by exploiting the structure of the code (compact hardware)
  - Independent of the chaining variable
Design Choices (3/3)

- Suitable for bitsliced implementation
Design Choices (3/3)

- Suitable for **bitsliced implementation**
  - Components from Serpent:
    - 4-bit Sbox, Linear Transformation L
Design Choices (3/3)

- Suitable for **bitsliced implementation**
  - Components from Serpent:
    4-bit Sbox, Linear Transformation L
- Concatenate-Permute-**Truncate**
Design Choices (3/3)

- Suitable for **bitsliced implementation**
  - Components from Serpent:
    - 4-bit Sbox, Linear Transformation L

- Concatenate-Permuate-Truncate
  - Expanded message **overwrites** part of the state → **Narrow-pipe**
Design Choices (3/3)

- Suitable for **bitsliced implementation**
  - Components from Serpent:
    - 4-bit Sbox, Linear Transformation L
- **Concatenate-Permute-Triminate**
  - Expanded message **overwrites** part of the state → **Narrow-pipe**
- Alternative option: **Concatenate-Permute-XOR**
Design Choices (3/3)

- Suitable for **bitsliced implementation**
  - Components from Serpent:
    - 4-bit Sbox, Linear Transformation L

- **Concatenate-Permuate-Truncate**
  - Expanded message *overwrites* part of the state $\rightarrow$ **Narrow-pipe**
  - Alternative option: **Concatenate-Permuate-XOR**
    - Expanded message is **XORed** into the state $\rightarrow$ **Wide-pipe**
General Design Approach
Security of Hamsi
Software/Hardware Performance
Conclusion

The Hash Function Hamsi

General Design

$M_1$

$M_2$

$M_i$
General Design

The Hash Function Hamsi
General Design Approach

Security of Hamsi

Software/Hardware Performance

Conclusion

General Design

The Hash Function Hamsi

Ozgur Kucuk

Katholieke Universiteit Leuven
General Design

\[ \begin{align*}
M_1 &\rightarrow h_0 \rightarrow \text{Concatenation} \rightarrow P \\
M_2 &\rightarrow h_1 \rightarrow \text{Concatenation} \rightarrow P \\
M_i &\rightarrow \oplus \rightarrow \text{Concatenation} \rightarrow P
\end{align*} \]
General Design

\[ M_1 \rightarrow h_0 \rightarrow \text{Concatenation} \rightarrow P \rightarrow h_1 \rightarrow \text{Concatenation} \rightarrow P \rightarrow \ldots \rightarrow M_2 \rightarrow \ldots \rightarrow M_i \]
General Design

\[ h_0 \rightarrow P \quad h_1 \rightarrow P \quad \cdots \cdots \quad h_l \rightarrow P_f \]

\[ M_1 \quad M_2 \quad M_i \]

\[ \oplus \quad \oplus \quad \cdots \cdots \quad \oplus \]

General Design Approach
Security of Hamsi
Software/Hardware Performance
Conclusion

Özgül Küçük
Katholieke Universiteit Leuven

The Hash Function Hamsi
General Design

\[ \begin{align*}
M_1 &\rightarrow h_0 \rightarrow \text{Concatenation} \rightarrow P \\
M_2 &\rightarrow h_1 \rightarrow \text{Concatenation} \rightarrow P \\
\ldots &\rightarrow h_{l-1} \rightarrow \text{Concatenation} \rightarrow P_f \\
&\rightarrow h
\end{align*} \]
Analysis
Analysis of the Compression Function

- “On the pseudorandomness of Hamsi,” J.P. Aumasson
- “Near Collisions for the Compression Function of Hamsi-256,” I. Nikolic
- “Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi-256,” J.P. Aumasson, W. Meier
- “Message Recovery and Pseudo-Preimage Attacks on the Compression Function of Hamsi-256,” Ç. Çalik, M.S. Turan
About Pseudo Near-Collisions
About Pseudo Near-Collisions

- Pseudo near-collisions are easy to construct
About Pseudo Near-Collisions

- Pseudo near-collisions are easy to construct
  - Not surprising
    Hamsi has a light compression function by design.
About Pseudo Near-Collisions

- Pseudo near-collisions are easy to construct
  - Not surprising
    Hamsi has a light compression function by design.
  - Of limited use for collision paths through hash function
    - Assumes that attacker has already obtained low weight $\Delta h_{i-1}$.
About Pseudo Near-Collisions

- Pseudo near-collisions are easy to construct
  - Not surprising
    Hamsi has a light compression function by design.
  - Of limited use for collision paths through hash function
    - Assumes that attacker has already obtained low weight $\Delta h_{i-1}$.
    - In all attacks, $\text{hw}(\Delta h_i)$ is greater than $\text{hw}(\Delta h_{i-1})$. 
About Pseudo Near-Collisions

- Pseudo near-collisions are easy to construct
  - Not surprising
    Hamsi has a light compression function by design.
  - Of limited use for collision paths through hash function
    - Assumes that attacker has already obtained low weight $\Delta h_{i-1}$.
    - In all attacks, $\text{hw}(\Delta h_i)$ is greater than $\text{hw}(\Delta h_{i-1})$.
    - Message expansion is bypassed by avoiding differences in the message.
About Pseudo Near-Collisions

- Pseudo near-collisions are easy to construct
  - Not surprising
    Hamsi has a light compression function by design.
- Of limited use for collision paths through hash function
  - Assumes that attacker has already obtained low weight $\Delta h_{i-1}$.
  - In all attacks, $\text{hw}(\Delta h_i)$ is greater than $\text{hw}(\Delta h_{i-1})$.
  - Message expansion is bypassed by avoiding differences in the message.
- Pseudo-collisions are much harder to construct.
Attack on the Hash Function
Attack on the Hash Function

Attack on the Hash Function

  - 2nd preimage attack on the hash function, $2^8$ times faster than generic attack, for messages of at least 8 blocks.
Attack on the Hash Function

  - 2nd preimage attack on the hash function, $2^8$ times faster than generic attack, for messages of at least 8 blocks.
  - Finding a 2nd preimage takes about $2^{248}$. 
Attack on the Hash Function

  - 2nd preimage attack on the hash function, $2^8$ times faster than generic attack, for messages of at least 8 blocks.
  - Finding a 2nd preimage takes about $2^{248}$.
  - Interesting and valid result, details are not published yet.
Attack on the Hash Function

  - 2nd preimage attack on the hash function, $2^8$ times faster than generic attack, for messages of at least 8 blocks.
  - Finding a 2nd preimage takes about $2^{248}$.
  - Interesting and valid result, details are not published yet.
- NIST: “Second-preimage resistance of approximately $n - k$ bits for any message shorter than $2^k$ bits.”
Attack on the Hash Function

  - 2nd preimage attack on the hash function, $2^8$ times faster than generic attack, for messages of at least 8 blocks.
  - Finding a 2nd preimage takes about $2^{248}$.
  - Interesting and valid result, details are not published yet.
- NIST: “Second-preimage resistance of approximately $n - k$ bits for any message shorter than $2^k$ bits.”
  - For 8-block messages (256 bits) NIST requires 248-bit security.
Attack on the Hash Function

  - 2nd preimage attack on the hash function, $2^8$ times faster than generic attack, for messages of at least 8 blocks.
  - Finding a 2nd preimage takes about $2^{248}$.
  - Interesting and valid result, details are not published yet.
- NIST: “Second-preimage resistance of approximately $n - k$ bits for any message shorter than $2^k$ bits.”
  - For 8-block messages (256 bits) NIST requires 248-bit security.
- Hamsi is a narrow-pipe design.
Attack on the Hash Function

  - 2nd preimage attack on the hash function, $2^8$ times faster than generic attack, for messages of at least 8 blocks.
  - Finding a 2nd preimage takes about $2^{248}$.
  - Interesting and valid result, details are not published yet.
- NIST: “Second-preimage resistance of approximately $n - k$ bits for any message shorter than $2^k$ bits.”
  - For 8-block messages (256 bits) NIST requires 248-bit security.
- Hamsi is a narrow-pipe design.
  - If first message is more than a few kilo bytes then there are faster generic attacks.
Performance
Software Performance
Software Performance

- Long messages:
  - 32cpb, Intel Core 2 Duo [eBASH].
  - 26cpb, Intel Core i7 [eBASH].
Software Performance

- **Long messages:**
  - 32cpb, Intel Core 2 Duo [eBASH].
  - 26cpb, Intel Core i7 [eBASH].

- **Short messages:**
  - 116cpb, Intel Core 2 Duo [eBASH].
  - 129cpb, Intel Core i7 [eBASH].
Software Performance

- **Long messages:**
  - 32cpb, Intel Core 2 Duo [eBASH].
  - 26cpb, Intel Core i7 [eBASH].

- **Short messages:**
  - 116cpb, Intel Core 2 Duo [eBASH].
  - 129cpb, Intel Core i7 [eBASH].

- Moderate speed for long messages.
- Among the best performers for short messages.
Hardware Performance
Hardware Performance

- Hamsi has a **small** state size.
  - 768-bit (including the feedforward).
Hardware Performance

- Hamsi has a small state size.
  - 768-bit (including the feedforward).

- As reported in many papers Hamsi has a good performance in FPGA and ASIC implementations.
Hardware Performance

- “Developing a Hardware Evaluation Method for SHA-3 Candidates,” Integrated Systems Laboratory of the ETH Zurich.
- “Fair and Comprehensive Methodology for Comparing Hardware Performance of Fourteen Round Two SHA-3 Candidates using FPGAs,” Kris Gaj et al.
- “Uniform Evaluation of Hardware Implementations of the Round-two SHA-3 Candidates,” S. Tillich et al.
Conclusion

- Hamsi has some unique design features.
- Received a fair amount of attention from cryptanalysts.
- It has attractive software/hardware performance.
More information:
[http://homes.esat.kuleuven.be/~okucuk/hamsi/]