

The Hash Function Hamsi

Özgül Küçük

Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC

February 27, 2009

Outline

Hamsi Specification

- General Design

- Message Expansion

- Hamsi-256

Design Choices and Analysis

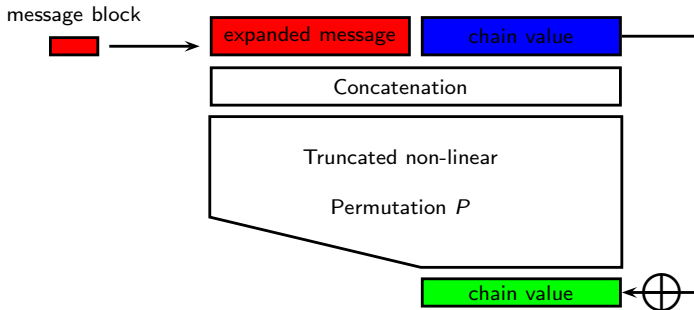
- How to Analyze Hamsi?

Implementation

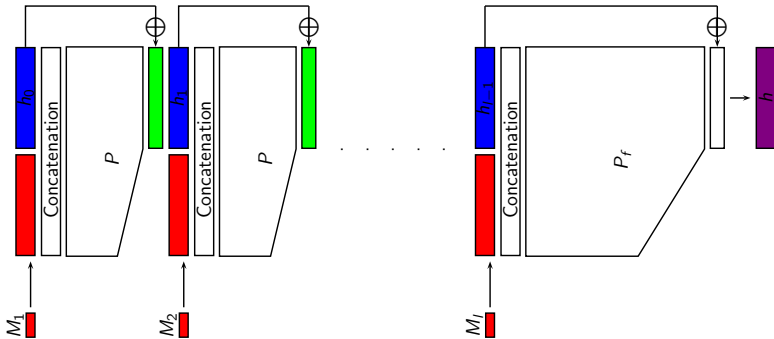
- Software-Hardware

Conclusion

General Design



General Design

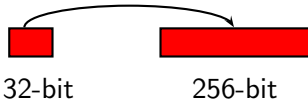


General Design

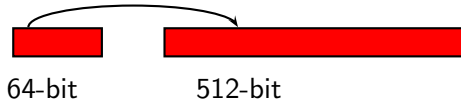
- ▶ General design is a sponge-like construction and influenced by Grindahl
 - ▶ Message expansion
 - ▶ Feedforward of the previous chaining value
 - ▶ 2 different non-linear permutations; P , P_f
- ▶ Hamsi iteration operates over small message blocks
 - ▶ 32-bit → Hamsi-256
 - ▶ 64-bit → Hamsi-512
 - ▶ Hamsi-256 → 512-bit internal state size
 - ▶ Hamsi-512 → 1024-bit internal state size

Message Expansion

Hamsi-256

Linear code $[128,16,70]$ over F_4 

Hamsi-512

Linear code $[256,32,131]$ over F_4 

Message Expansion

- ▶ **B**est **K**nown **L**inear **C**odes; optimal choice for given parameters
- ▶ Minimum distance → active Sboxes for one round
 - ▶ **70** for Hamsi-256
 - ▶ **131** for Hamsi-512
- ▶ Provides **strong diffusion** against differential type attacks
- ▶ Can be computed in parallel
- ▶ Can be implemented efficiently in software and hardware

Permutations P and P_f

- ▶ XOR of constants and a counter
- ▶ Substitution is by 4×4 -bit Sboxes
- ▶ Diffusion is by a linear transformation L

- ▶ Sbox and L are from the block cipher Serpent
- ▶ P and P_f only differ in the number of rounds and constants
 - ▶ P has 3 rounds, P_f has 6 rounds
 - ▶ P_f is only applied at the finalization

Concatenation

$(m_0, m_1, \dots, m_7, c_0, c_1, \dots, c_7)$ \xrightarrow{C}

Hamsi-256 state

m_0	m_1	c_0	c_1
c_2	c_3	m_2	m_3
m_4	m_5	c_4	c_5
c_6	c_7	m_6	m_7

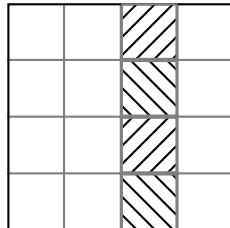
- ▶ Each Sbox inputs 2-bits from the chain value and the expanded message.
- ▶ Chaining value and expanded message words appear in all bit positions of the Sboxes.

Substitution

Hamsi-256 state

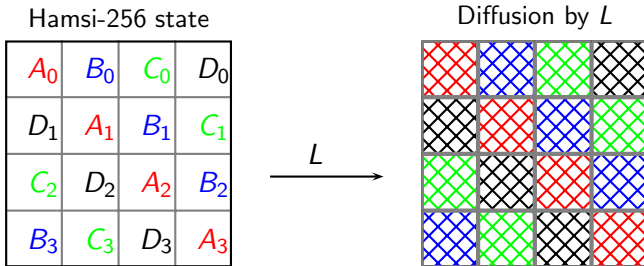
s_0	s_1		s_3
s_4	s_5		s_7
s_8	s_9		s_{11}
s_{12}	s_{13}		s_{15}

S



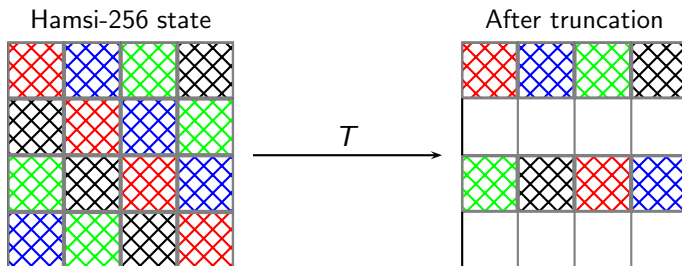
- ▶ 4×4 *Serpent* Sbox
- ▶ Confusion/diffusion across the rows
- ▶ Can be processed in parallel, word size up to 128-bits
- ▶ Suitable for bitsliced implementation

Diffusion



- ▶ L consists of XOR, rotation and shift operations
- ▶ L operates on 32-bit words; inputs and outputs 4 words
- ▶ Any single bit difference amplified by L affects a different Sbox

Truncation



Design Choices and Analysis

- ▶ Design choices are due to preliminary analysis
 - ▶ Strong message expansion
 - ▶ Concatenation can be viewed as an initial mixing
 - ▶ Sboxes have good differential/linear and avalanche properties
 - ▶ Algebraic degree of all but one output bit is 3; we are not aware of any attacks based on this.
 - ▶ Feedforward is added against generic preimage attacks
 - ▶ L is applied in such a way to destroy symmetry
 - ▶ P_f is to prevent against length extension and slide attacks

How to break Hamsi?

- ▶ Internal collisions
 - ▶ Any difference on the message very quickly activates many Sboxes (thanks to message expansion...)
 - ▶ Message modification can not help to cancel difference propagation (message block is concatenated to the chain value)
- ▶ External collision seems much harder to find... (P_f has 6 rounds)

How to break Hamsi?

- ▶ Analyze the following:
 - ▶ Apply a difference **only** on the chain value
 - ▶ Propagate as far as you find a **collision**
 - ▶ Work backwards to find a matching difference on the message
 - ▶ Show that overall complexity is **lower** than the birthday bound
- ▶ break 1 round of Hamsi-256; appreciated! (easy...?)
- ▶ break 2 rounds of Hamsi-256; very much appreciated!
- ▶ break 3 rounds of Hamsi-256; (well... will be good to know...)
- ▶ Use your imagination!

Software-Hardware

- ▶ Hamsi is a software and hardware friendly algorithm
- ▶ Hamsi-256 runs at **25cpb** on Intel Core2 (in C using SSE2 instructions)
 - ▶ Further improvements are in progress
- ▶ Hardware results for Hamsi-256:
 - ▶ Area: **14.5KGate**, UMC 130 nm library, typical case
 - ▶ Timing: 3.2ns as critical path delay, 310MHz
 - ▶ Throughput: **3.3Gbps**

Conclusion

- ▶ Hamsi is a secure and fast hash function
- ▶ Feel **free** to analyze it
- ▶ Feel **free** to make it faster

- ▶ `http://homes.esat.kuleuven.be/~okucuk/hamsi/`