SHAvite-3: Secure, Efficient, and Flexible Hash Function Proposal

Orr Dunkelman

Département d'Informatique
École Normale Supérieure

France Telecom Chaire

Joint work with Eli Biham
Outline

1 Specification and Design Rationale
   - SHA\textit{v}ite-3
   - SHA\textit{v}ite-3\textsubscript{256} — Producing Digests of up to 256 Bits
   - The $E^{256}$ Block Cipher
   - SHA\textit{v}ite-3\textsubscript{512} — Producing Digests of 257 to 512 Bits
   - The $E^{512}$ Block Cipher
   - Why SHA\textit{v}ite-3?

2 Security Analysis
   - The Security of the Block Ciphers
   - The Security as a Hash Function
   - Theoretical Notions of Security

3 Performance Results and Analysis
   - Software Implementation
   - Hardware Implementation

4 The SHA\textit{v}ite-3-MAC
   - Definition of the SHA\textit{v}ite-3-MAC
   - Comparing SHA\textit{v}ite-3-MAC with HMAC
Outline

1 Specification and Design Rationale
   - SHAvite-3
   - SHAvite-3_{256} — Producing Digests of up to 256 Bits
   - The $E^{256}$ Block Cipher
   - SHAvite-3_{512} — Producing Digests of 257 to 512 Bits
   - The $E^{512}$ Block Cipher
   - Why SHAvite-3?

2 Security Analysis
   - The Security of the Block Ciphers
   - The Security as a Hash Function
   - Theoretical Notions of Security

3 Performance Results and Analysis
   - Software Implementation
   - Hardware Implementation

4 The SHAvite-3-MAC
   - Definition of the SHAvite-3-MAC
   - Comparing SHAvite-3-MAC with HMAC
SHA\v{v}ite-3 (SHA\v{v}ite-Shalosh)

- A SHA-3 candidate designed to be secure, efficient, and suitable for all environments.
- SHA\v{v}ite-3_{256} is used for digests up to 256 bits, and SHA\v{v}ite-3_{512} is used for digests of 257 to 512 bits.
- The compression functions are iterated using HAIFA.
- Supports salts (nonces/randomized hashing), variable digest length, while maintaining full security.
- The compression function is designed using known and understood components: Feistel structure, AES-round, and LFSRs.
Based on the $C_{256}$ compression function,
Based on the $C_{256}$ compression function,

- which is a Davies-Meyer transformation of the block cipher $E^{256}$,
Based on the $C_{256}$ compression function,
  - which is a Davies-Meyer transformation of the block cipher $E_{256}$,
    - which is a 12-round Feistel block cipher,
Based on the $C_{256}$ compression function,

- which is a Davies-Meyer transformation of the block cipher $E_{256}$,
  - which is a 12-round Feistel block cipher,
    - where each round function is composed of three AES rounds.
    - The message expansion combines both AES rounds and LFSRs.
Advanced Encryption Standard

- AES was selected at the end of a similar process to the SHA-3 process by NIST in 2000.
- The selected algorithm, Rijndael, was selected from 15 submissions, of which 5 became known the AES finalists.
- Thoroughly analyzed in many cryptographic settings, and so far withstood all cryptanalytic attempts.
- Best known attack: 7/10 rounds for 128-bit keys, 8/12 rounds for 192-bit, and 8/14 rounds for 256-bit keys (in the related-key model, the results are 7/9/10 rounds, respectively).
$E^{256}$ — the Underlying Block Cipher

- Accepts a 256-bit plaintext (chaining value).
- Accepts a key (message block, bit counter, and a salt) of 832 bits in total.
- The round function is composed of 3 rounds of AES (with an AddRoundKey operation before the first round, last AddRoundKey operation omitted).
- The message expansion generates 36 128-bit subkeys (12 rounds of $E^{256}$, each uses 3 round of AES).
The Underlying Block Cipher (cont.)

\[ E^{256} \]
The Message Expansion ($E^{256}$ Key Schedule)

- Accepts 832-bit key: 512-bit block, 256-bit salt, 64-bit counter.
- Not all bits are treated equally.
- A combination of an LFSR (for diffusion), and AES rounds (for maximal “confusion” and nonlinearity).
The Message Expansion ($E^{256}$ Key Schedule) (cont.)
The Message Expansion ($E^{256}$ Key Schedule) (cont.)

- The key (message block, counter, and salt) is expanded into 144 32-bit words.
- The first 16 32-bit words are the message words.
- The following process is repeated four times:
  1. 16 words are generated by applying AES round (where the salt is XORed before the round) and some XORs.
  2. 16 words are generated using an LFSR operation.
- The counter words are mixed into 8 of the 144 words, to ensure that the counter affects the encryption process.
Final Touches — SHA\textit{v}i\textit{t}e-3\textsubscript{256}

- In order to hash the message $M$ into an $m$-bit digest, for $m \leq 256$, compute $IV_m$ which is

$$h_0 = IV_m = C_{256}(MIV_{256}, m, 0, 0),$$

- Let $|M|$ be the length of $M$ before padding, measured in bits. Pad the message $M$ according to the padding scheme of HAIFA:
  
  1. Pad a single bit of 1.
  2. 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 512 to 432.
  3. Pad $|M|$ encoded in 64 bits.
  4. Pad $m$ encoded in 16 bits.

- Divide the padded message $pad(M)$ into 512-bit blocks,

$$pad(M) = M_1\|M_2\|\ldots\|M_l,$$
Final Touches — SHA\textit{v}i\textit{t}e-3_{256}

- Set \#\textit{bits} ← 0.
- Set \(h_0 \leftarrow IV_m\).
- For \(i = 1, \ldots, \lfloor |M|/512 \rfloor\):
  - Set \#\textit{bits} ← \#\textit{bits} + 512.
  - Compute \(h_i = C_{256}(h_{i-1}, M_i, \#\textit{bits}, salt)\).
  - If \(|M| = 0 \mod 512\), compute
    \(h_l = C_{256}(h_{l-1}, M_l, 0, salt)\), else
  - If \(|M| \mod 512 \leq 431\), compute
    \(h_l = C_{256}(h_{l-1}, M_l, |M|, salt)\), else
  - Compute \(h_{l-1} = C_{256}(h_{l-2}, M_{l-1}, |M|, salt)\), and then
    compute \(h_l = C_{256}(h_{l-1}, M_l, 0, salt)\).
- Output \(\text{truncate}_m(h_l)\), where \(\text{truncate}_m(x)\) outputs the \(m\) leftmost bits of \(x\), i.e., \(x[0]\|x[1]\|\ldots\).
SHA\textit{v}ite-3_{512}

Based on the $C_{512}$ compression function,
Based on the $C_{512}$ compression function,

- which is a Davies-Meyer transformation of the block cipher $E^{512}$,
Based on the $C_{512}$ compression function, which is a Davies-Meyer transformation of the block cipher $E^{512}$, which is a 14-round Generalized Feistel block cipher,
Based on the $C_{512}$ compression function,

- which is a Davies-Meyer transformation of the block cipher $E^{512}$,
  - which is a 14-round Generalized Feistel block cipher,
    - where each round function is composed of four AES rounds.
$E^{512}$ — the Underlying Block Cipher

- Accepts a 512-bit plaintext (chaining value), and 1664-bit key (message block, counter, and salt).
- The block cipher has a Generalized Feistel structure.
- The plaintext is divided into four words of 128 bits each.
- In each of the 14 rounds, two words enter (separately) the round function.
- After XORing the output of the round function with the two remaining words, the words are rotated.
$E^{512}$ — the Underlying Block Cipher (cont.)
How to Pronounce SHA-vite-3

SHA-vite SHA-losh
What Does it Mean?

- SHA + vite — fast secure hash algorithm.
What Does it Mean?

- SHA + vite — fast secure hash algorithm.
- shavit means “comet” in Hebrew
What Does it Mean?

- SHA + vite — fast secure hash algorithm.
- shavit means “comet” in Hebrew
- A follower of Shiva, god of destruction, is called Shavite.
What Does it Mean?

- SHA + vite — fast secure hash algorithm.
- shavit means “comet” in Hebrew
- A follower of Shiva, god of destruction, is called Shavite.
- shalosh means 3 in Hebrew.
Outline

1 Specification and Design Rationale
   - SHAvite-3
   - SHAvite-3\textsubscript{256} — Producing Digests of up to 256 Bits
   - The $E^{256}$ Block Cipher
   - SHAvite-3\textsubscript{512} — Producing Digests of 257 to 512 Bits
   - The $E^{512}$ Block Cipher
   - Why SHAvite-3?

2 Security Analysis
   - The Security of the Block Ciphers
   - The Security as a Hash Function
   - Theoretical Notions of Security

3 Performance Results and Analysis
   - Software Implementation
   - Hardware Implementation

4 The SHAvite-3-MAC
   - Definition of the SHAvite-3-MAC
   - Comparing SHAvite-3-MAC with HMAC
Is the Block Cipher $E^{256}$ Secure?

- Of course!
Is the Block Cipher $E^{256}$ Secure?

- Of course!
- The maximal expected differential probability of three round AES is at most $2^{-49}$.
- Analysis reveals that there are no 2-round iterative characteristics, 3-round iterative characteristics of probability higher than $2^{-98}$, ..., 9-round characteristics with probability higher than $2^{-294}$.
- Similar results hold for linear cryptanalysis/boomerang attacks.
- Longest known impossible differential is of 5 rounds.
- Longest known SQUARE is of 3 rounds.
- Slide/Related-key attacks — counter protects against these.
- Algebraic attacks: equations reach full degree after 4 rounds.
Is the Block Cipher $E^{512}$ Secure?

- The maximal expected differential probability of four round AES is at most $2^{-113}$.
- Analysis reveals that there are no 2-round iterative characteristics, 3-round iterative characteristics of probability higher than $2^{-113}$, ... , 9-round characteristics with probability higher than $2^{-678}$.
- Similar results hold for linear cryptanalysis/boomerang attacks.
- Longest known impossible differential is of 9 rounds.
- Longest known SQUARE is of 3 rounds.
- Slide/Related-key attacks — counter protects against these.
- Algebraic attacks: equations reach full degree after 4 rounds.
Extending the Block Cipher Security Results

First Attempt

Computing the maximal expected differential probability of related-key attacks.
Extending the Block Cipher Security Results

First Attempt

Computing the maximal expected differential probability of related-key attacks.

First Attempt Fails

No good methodology for that.
# Extending the Block Cipher Security Results

**First Attempt**

Computing the maximal expected differential probability of related-key attacks.

**First Attempt Fails**

No good methodology for that.

**First Attempt Fails (2)**

The attacker controls the keys, he can make sure some differential transitions do happen.
Extending the Security Results — 2nd Attempt

What to do

Consider differentials through the message expansion.

- For a fixed salt, the message expansion can be treated as a block cipher.
- Compute the probability of differentials of it.
- Count how many active S-boxes there are in the message expansion.
- Assume that the attacker can use message modification to increase the probability of the differential (fixing 8 bits of the message/salt/counter can “eliminate” the cost of one active S-box).
- Results: No good differentials. The message expansion makes sure there are no high probability differentials through the message expansion.
A 3rd Attempt (TBD)

- Collision-producing differentials need to go both through the message expansion and the block cipher.
- Each probabilistic event in any of the two should “cost”:
  1. Each active byte that enters the actual compression data-path costs at least 8 bits of control.
  2. Each transition of difference column through the MDS matrix in the message expansion — costs control according to the hamming weights.
  3. Each XOR in the message expansion that cancels a difference — costs control.
- Too large of a search space, but gives a very strong upper bound.
HAIFA offers a prefix-free encoding:

- If the compression function is a random oracle the hash function is indifferentiable from random oracle (up to the birthday bound).
- Maintaining the salt secret leads to an efficient and secure PRF (MAC).
- If the compression function is a random oracle, the second preimage resistance can be proved to be $O(2^n)$. 
Outline

1. Specification and Design Rationale
   - SHA\textit{v}ite-3
   - SHA\textit{v}ite-3\textsubscript{256} — Producing Digests of up to 256 Bits
   - The \(E^{256}\) Block Cipher
   - SHA\textit{v}ite-3\textsubscript{512} — Producing Digests of 257 to 512 Bits
   - The \(E^{512}\) Block Cipher
   - Why SHA\textit{v}ite-3?

2. Security Analysis
   - The Security of the Block Ciphers
   - The Security as a Hash Function
   - Theoretical Notions of Security

3. Performance Results and Analysis
   - Software Implementation
   - Hardware Implementation

4. The SHA\textit{v}ite-3-MAC
   - Definition of the SHA\textit{v}ite-3-MAC
   - Comparing SHA\textit{v}ite-3-MAC with HMAC
# Software Implementation

<table>
<thead>
<tr>
<th>Hash Function</th>
<th>32-Bit</th>
<th>64-Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>7.4</td>
<td>8.8</td>
</tr>
<tr>
<td>SHA-1</td>
<td>9.8</td>
<td>9.5</td>
</tr>
<tr>
<td>SHA-256</td>
<td>28.8</td>
<td>25.3</td>
</tr>
<tr>
<td>SHA-512</td>
<td>77.8</td>
<td>16.9</td>
</tr>
<tr>
<td>SHAvite-3&lt;sub&gt;256&lt;/sub&gt; (measured)</td>
<td>35.3</td>
<td>26.7</td>
</tr>
<tr>
<td>SHAvite-3&lt;sub&gt;256&lt;/sub&gt; (conjectured)</td>
<td>26.6</td>
<td>18.6</td>
</tr>
<tr>
<td>SHAvite-3&lt;sub&gt;256&lt;/sub&gt; (with AES inst.)</td>
<td>&lt; 8</td>
<td></td>
</tr>
<tr>
<td>SHAvite-3&lt;sub&gt;512&lt;/sub&gt; (measured)</td>
<td>55.0</td>
<td>38.2</td>
</tr>
<tr>
<td>SHAvite-3&lt;sub&gt;512&lt;/sub&gt; (conjectured)</td>
<td>35.3</td>
<td>28.4</td>
</tr>
<tr>
<td>SHAvite-3&lt;sub&gt;512&lt;/sub&gt; (with AES inst.)</td>
<td>&lt; 12</td>
<td></td>
</tr>
</tbody>
</table>

Expect 1–1.5 cycles per byte improvement if no salts are used.
We looked at four hardware optimizations for AES: FPGA/ASIC, fastest/smallest.

<table>
<thead>
<tr>
<th>Digest Size</th>
<th>Technology</th>
<th>Size</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>ASIC</td>
<td>10.3 Kgates</td>
<td>7.6 Mbps</td>
</tr>
<tr>
<td></td>
<td>FPGA</td>
<td>55.0 Kgates</td>
<td>604.4 Mbps</td>
</tr>
<tr>
<td>512</td>
<td>ASIC</td>
<td>18.5 Kgates</td>
<td>4.7 Mbps</td>
</tr>
<tr>
<td></td>
<td>FPGA</td>
<td>81 Kgates</td>
<td>907.7 Mbps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3585 Slices</td>
<td>872.3 Mbps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>895 Slices</td>
<td>1.12 Gbps</td>
</tr>
</tbody>
</table>

These are estimates based on AES implementations from 2005. Expect real figures to be much better.
Outline

1 Specification and Design Rationale
   - SHA\textit{v}ite-3
   - SHA\textit{v}ite-3\textsubscript{256} — Producing Digests of up to 256 Bits
   - The $E^{256}$ Block Cipher
   - SHA\textit{v}ite-3\textsubscript{512} — Producing Digests of 257 to 512 Bits
   - The $E^{512}$ Block Cipher
   - Why SHA\textit{v}ite-3?

2 Security Analysis
   - The Security of the Block Ciphers
   - The Security as a Hash Function
   - Theoretical Notions of Security

3 Performance Results and Analysis
   - Software Implementation
   - Hardware Implementation

4 The SHA\textit{v}ite-3-MAC
   - Definition of the SHA\textit{v}ite-3-MAC
   - Comparing SHA\textit{v}ite-3-MAC with HMAC
The SHA\textit{v}ite-3-MAC

- With HAIFA, one can define

\[ \text{HAIFA-MAC}^C_k(M) = \text{HAIFA}^C_k(M). \]

- As HAIFA is PRF-preserving, then the above MAC is secure.
The SHA\textit{v}ite-3-MAC

- With HAIFA, one can define

$$\text{HAIFA-MAC}_k^C(M) = \text{HAIFA}_k^C(M).$$

- As HAIFA is PRF-preserving, then the above MAC is secure.

- SHA\textit{v}ite-3 is secure, and thus we define

$$\text{SHA}\text{v}ite-3\text{-MAC}_k(M) = \text{SHA}\text{v}ite-3_k(M).$$

- Of course, the user needs to keep the key secret!
Comparison with HMAC

- More efficient — most of the time, one compression function less than HMAC.
- More efficient — one less initialization than HMAC.
- Better foundations for the security analysis.

Number of compression function calls:

<table>
<thead>
<tr>
<th>Construction</th>
<th>0 Bytes</th>
<th>1500 Bytes</th>
<th>$n$ Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-256</td>
<td>1</td>
<td>24</td>
<td>$\lceil (n + 8)/64 \rceil$</td>
</tr>
<tr>
<td>HMAC-SHA-256</td>
<td>2</td>
<td>25</td>
<td>$1 + \lceil (n + 8)/64 \rceil$</td>
</tr>
<tr>
<td>SHAvite-3</td>
<td>1</td>
<td>24</td>
<td>$\lceil (n + 10)/64 \rceil$</td>
</tr>
<tr>
<td>SHAvite-3-MAC</td>
<td>1</td>
<td>24</td>
<td>$\lceil (n + 10)/64 \rceil$</td>
</tr>
</tbody>
</table>
Thank you for your attention!

http://www.cs.technion.ac.il/~orrd/SHAvite-3/