Minalpher

DIAC 2014

Yu Sasaki\textsuperscript{1}, Yosuke Todo\textsuperscript{1}, Kazumaro Aoki\textsuperscript{1}, Yusuke Naito\textsuperscript{2}, Takeshi Sugawara\textsuperscript{2}, Yumiko Murakami\textsuperscript{2}, \textbf{Mitsuru Matsui}\textsuperscript{2}, Shoichi Hirose\textsuperscript{3}

\textsuperscript{1}: NTT  \textsuperscript{2}: Mitsubishi Electric  \textsuperscript{3}: Fukui University
How to pronounce

Minalpher [mɪnˈælfə]

Alpha → Alpher → Min-alpher → Minalpher
Minalpher is already a winner in the categories of...

Longest Name: 9 chars
with AVALANCHE, Enchilada and Raviyoyla

Longest Document: 70 pages
we designed everything from scratch
Minalpher: Design Concepts
Easy to Use in Practice

• 128-bit security (with 256-bit permutation)
  – 128-bit confidentiality
  – 128-bit authenticity

• Additional security in misuse scenarios
  – nonce repetition (nonce misuse)
  – release of unverified plaintext (decryption misuse)
Minalpher: Design Concepts
Easy to Use in Practice

• One algorithm for various platforms
  – Fully parallelizable mode (fast on high-end platforms)
  – Simple, repetitive structure (small on embedded systems)

• Additional functionalities
  – MAC-only mode (faster than ciphertext-discard AEAD)
  – Associated data reuse (faster 2\textsuperscript{nd} time and afterwards)
  – Incremental computation (faster in nonce reuse scenario)

• No Patent Submitted
DESIGN
Minalpher (AEAD mode): Overview

\[ K || \text{flag}_{ad} || 0^{n/2-s} \]

\[ K || \text{flag}_m || N \]

\[ P \]

\[ L' \]

\[ L \]

\[ \gamma_1 \]

\[ \gamma_{a-1} \]

\[ \gamma_a \text{ or } \gamma''_a \]

\[ \psi_1 \]

\[ \psi_{m-1} \text{ or } \psi_m \]

\[ \psi'_m \]

\[ \psi' \]

\[ \varphi_1 \]

\[ \varphi_{m-1} \text{ or } \varphi_m \]

\[ \varphi_i = y^{2i-1} L \]

\[ \psi_i = y^{2i} L \]

\[ \psi_m = y^{2m-1} (y+1) L \]

\[ \psi_m' \]

\[ \gamma_i = y^i L' \]

\[ \gamma_a = y^{a-1} (y+1) L' \]

\[ \gamma''_a = y^{a-1} (y+1)^2 L' \]
Minalpher (MAC mode): Overview

\[ y \] denotes a root of
\[ Y^{32} + Y^3 + Y^2 + x = 0, \]
where \( x \) is a root of
\[ X^8 + X^7 + X^5 + X + 1 = 0 \]
Design Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Size</td>
<td>128 bits</td>
</tr>
<tr>
<td>Nonce Size</td>
<td>104 bits</td>
</tr>
<tr>
<td>Tag Size</td>
<td>128 bits</td>
</tr>
<tr>
<td>Block Size</td>
<td>256 bits</td>
</tr>
</tbody>
</table>

Max Size of AD+MSG in the AEAD mode
2^{104} – 1 bits

Max Size of MSG in the MAC mode
2^{104} – 1 bits

(secret message number is not supported in Minalpher)
Minalpher-P: 256-bit Permutation

• Nibble-wise (4-bit) Architecture
  – 1 block = 256 bits = 64 nibbles

• 17.5-round Involutiue SPN Structure
  – $SN$ (SubNibble): An Involutiue 4-bit S-box
  – $SR$ (ShuffleRows): Byte shuffle + Nibble swap
  – $MC$ (MixColumns): A Binary 4x4 Matrix

• $P = P^{-1}$ except round constants
Minalpher-P: One Round

$E(i-1) \rightarrow X_i$

$X_{i-1} \rightarrow S \rightarrow T \rightarrow M \rightarrow X_i$

$A_{i-1}$

$A^{SN}_{i-1} \rightarrow A^{SR}_{i-1}$

$B^{SN}_{i-1} \rightarrow B^{SR}_{i-1}$

$A^{SM}_{i-1} \rightarrow A^{XM}_{i-1}$

$B^{SM}_{i-1} \rightarrow B^{XM}_{i-1}$

$A^M_{i-1}$

$B^M_{i-1}$

$X_{i-1}$

$X_i$

$A_i$

$B_i$

$RC_{i-1}$

$SR_1$

$SR_2$

$1 1 0 1$

$1 1 1 0$

$0 1 1 1$

$1 0 1 1$
SECURITY
Security of Tweakable Even-Mansor

• Tweakable Even-Mansor (TEM) is a 128-bit Strong Tweakable Pseudorandom Permutation (STPRP) in the ideal permutation model.

\[
\text{Adv}_{\text{TEM}}^{\text{STPRP}}(D) = \Pr \left[ K \xleftarrow{\text{R}} \mathcal{K} : D_{\text{TEM}_\text{Enc}_K,\text{TEM}_\text{Dec}_K,P,P^{-1}} \Rightarrow 1 \right] - \Pr \left[ D_{\text{TRP}_F,\text{TRP}_B,P,P^{-1}} \Rightarrow 1 \right] \\
\leq \frac{\sigma^2}{2^{n-1}} + \frac{\sigma}{2^{n/2}}
\]

Tweakable Even-Mansor Encryption with Minalpher -P
Security of Mode of Operation

• Minalpher achieves 128-bit security for both privacy and authenticity

\[ \text{Adv}^{\text{priv}}(A) = \Pr[K \leftarrow K : A^{E_K} \Rightarrow 1] - \Pr[A^S \Rightarrow 1] \]
\[ \leq \text{Adv}^{\text{tprp}}_{\text{TEM}}(D) + \frac{\sigma^2}{2^{n+1}} + \frac{q^2}{2^n} \]

• Authenticity: INT-CCA

\[ \text{Adv}^{\text{auth}}(A) = \Pr[K \leftarrow K : A^{E_K, D_K} \text{ forges}] \]
\[ \leq \text{Adv}^{\text{stprp}}_{\text{TEM}}(D) + \frac{q}{2^{l}} + \frac{\sigma^2}{2^{n+1}} + \frac{q^2}{2^n} \]
Security in Misuse Scenarios

- Minalpher achieves full authenticity and some privacy even in the following **misuse** scenarios:
  - Release of unverified plaintext (RUP), Nonce repetition (NR)

<table>
<thead>
<tr>
<th>Authenticity</th>
<th>Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Authenticity Table" /></td>
<td><img src="image" alt="Privacy Table" /></td>
</tr>
</tbody>
</table>

**BW-PRF**: Block-wise pseudo-random-function

Blocks $ct[i]$ are indistinguishable from $RF(K, N, A, pt[i])$
Security against Various Cryptanalysis

• 128-bit key + 256-bit block
  – The structure prevents a class of cryptanalysis requiring at least $2^{128}$ cost, e.g. MitM attacks, rectangle attacks.

• Enough Security Margin
  – Differential/linear characteristic probability at most $2^{-128}$ in 7 rounds out of the full 17.5 Minalpher-P rounds.
  – No 12-round attacks of Minalpher(-P) detected so far, e.g. boomerang attacks, amplified boomerang attacks, integral attacks, impossible differential attacks, truncated differential attacks, rebound attacks, etc.
PERFORMANCE
Hardware implementation

• Small S-box (4x4) and regular structure enable efficient and scalable Minalpher-P circuits
  – No need for a key-scheduling circuit
  – Small S-box based design common in lightweight crypto
  – Involutive property minimizes the number of selectors

• Three different hardware architectures are shown in the document
  – High-speed core, mid-range core and low-area coprocessor
  – Evaluated with an open-source library: NanGate 45-nm CMOS

• Further high throughput is possible if parallelized
**High-Speed Core**

<table>
<thead>
<tr>
<th></th>
<th>Area [kGE]</th>
<th>Throughput [Mbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minalpher-P Enc. and Dec., 1-round/cycle</td>
<td>4.96</td>
<td>7,771.71</td>
</tr>
<tr>
<td>AES Enc. Only, 1-round/cycle</td>
<td>10.49</td>
<td>1,587.50</td>
</tr>
<tr>
<td>Minalpher</td>
<td>14.32</td>
<td>6,103.96</td>
</tr>
</tbody>
</table>

- **No selector at the main path**
- **Very small constant generator**
- **Fast & small 4-bit S-box**
- **High-speed core**

**Data in**

**Counter**

**Const gen.**

**S**

**T**

**M**

**Data out**

**Sequencer**
Low-Area Coprocessor

<table>
<thead>
<tr>
<th></th>
<th>Area [kGE]</th>
<th>Throughput [Mbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minalpher-P, Enc. and Dec., 16-bit datapath</td>
<td>2.70</td>
<td>375.06</td>
</tr>
<tr>
<td>AES Enc. only, 8-bit datapath</td>
<td>3.71</td>
<td>50.52</td>
</tr>
<tr>
<td>Minalpher</td>
<td>2.81</td>
<td>369.34</td>
</tr>
</tbody>
</table>

Almost no selectors at the main path

Fast & Small 4-bit S-box (again)

Complexity of ShuffleRows can be absorbed in the shift-register layer
Minalpher on Intel 64 Architecture

• Minalpher is designed to be well-suited on Intel 64 platform.
• The vpshufb instruction works very efficiently on SN (SubNibbles) and SR (ShuffleRows).
• Parallel block implementation can achieve faster speed.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Implementation Method</th>
<th>Data Length / Cycles per byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>31B</td>
</tr>
<tr>
<td>Core i7-3770 (Ivy Bridge)</td>
<td>1-block</td>
<td>23.1</td>
</tr>
<tr>
<td>Core i7-3770 (Ivy Bridge)</td>
<td>2-block parallel</td>
<td>23.4</td>
</tr>
<tr>
<td>Core i7-4770 (Haswell)</td>
<td>4-block parallel</td>
<td>---</td>
</tr>
</tbody>
</table>

*Estimation based on the implementation of Minalpher-P
Round Function on Ivy/Sandy Bridge (1-block implementation)

Each nibble is stored in an octet of an XMM register. Each register contains both rows of A and B.

XMM Register
A Code Example of One Round

State-In

(vpshufb xmm0, xmm15, xmm0)
(vpshufb xmm1, xmm15, xmm1)
(vpshufb xmm2, xmm15, xmm2)
(vpshufb xmm3, xmm15, xmm3)
(vpshufb xmm0, xmm0, xmm14)
(vpshufb xmm1, xmm1, xmm13)
(vpshufb xmm2, xmm2, xmm12)
(vpshufb xmm3, xmm3, xmm11)
(vpslldq xmm4, xmm0, 8)
(vpslldq xmm5, xmm1, 8)
(vpslldq xmm6, xmm2, 8)
(vpslldq xmm7, xmm3, 8)
(pxor xmm4, xmm0)
(pxor xmm5, xmm1)
(pxor xmm6, xmm2)
(pxor xmm7, xmm3)

State-Out

(vpshufb xmm15, xmm0, xmm0)
(vpshufb xmm15, xmm1, xmm1)
(vpshufb xmm15, xmm2, xmm2)
(vpshufb xmm15, xmm3, xmm3)
(vpslldq xmm15, xmm0, 8)
(vpslldq xmm15, xmm1, 8)
(vpslldq xmm15, xmm2, 8)
(vpslldq xmm15, xmm3, 8)
(pxor xmm15, xmm0)
(pxor xmm15, xmm1)
(pxor xmm15, xmm2)
(pxor xmm15, xmm3)

S-box table is stored in xmm15
SR table is stored in xmm11-14
Minalpher on Low-end Microcontrollers

• Minalpher is designed to be implemented in a small footprint on low-end microcontrollers
  – The architecture Minalpher(-P) is simple and repetitive
  – A single involutional 4-bit S-box
  – Multiplying “y” in the tweak update is simple (3 byte-xors and x*2 in GF($2^8$))

• High speed implementation is also possible
  – Two adjacent 4-bit S-boxes can be regarded as an 8-bit lookup table, which significantly improves performance.
## Implementation Results

- **Target processor:** RL78 (CISC microcontroller)
- **Two Implementations**
  - **Small:** minimizing ROM size
  - **Fast:** maximizing speed

<table>
<thead>
<tr>
<th>Design Goal</th>
<th>ROM (Bytes)</th>
<th>RAM (Bytes)</th>
<th>Speed (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Init</td>
<td>AD</td>
<td>Enc / Dec</td>
</tr>
<tr>
<td>Small</td>
<td>510</td>
<td>214</td>
<td>90,235 45,302 90,992/91,081</td>
</tr>
<tr>
<td>Fast</td>
<td>1,275</td>
<td>470</td>
<td>16,805 8,166 16,447/16,669</td>
</tr>
</tbody>
</table>

- **Init:** Initialization (computing L and L’)
- **AD:** Processing of an associate data block
- **Enc/Dec:** Processing of an encryption/decryption block
Conclusions

Minalpher: Easy to use with simple and unique design
- 128-bit security for privacy and authenticity
- Supporting MAC mode
- Security in misuse scenarios
- Fixed associate data reuse
- Incremental AE/MAC
- Fully parallelizable
- Lightweight tweak generation
- Involutive permutation
- Small S-box
Thank you!