Updates on Romulus, Remus and TGIF

Tetsu Iwata\textsuperscript{1}  Mustafa Khairallah\textsuperscript{2}
Kazuhiko Minematsu\textsuperscript{3}  Thomas Peyrin\textsuperscript{2}

Nagoya University\textsuperscript{1}  Nanyang Technological University\textsuperscript{2}  NEC\textsuperscript{3}

NIST Third Lightweight Cryptography Workshop, USA
November 4-6, 2019
Romulus, Remus, and TGIF

Romulus
- A TBC-based AEAD mode
- Standard model security
- Skinny [BJK+16] as Tweakable Block Cipher

Remus
- An aggressively optimized version of Romulus
- Ideal-Cipher model security
- Skinny as Block Cipher (or IC)

TGIF
- Remus with a new cipher based on GIFT [BPP+17]
  - Designers: Yu Sasaki, Siang Meng Sim, Ling Sun and Romulus/Remus team

This talk’s focus: Romulus, as a 2nd-round candidate
Our Updates

Security

- Improved Security Bounds
- No dependency on the input length, in most cases

Implementation

- Hardware (ASIC and FPGA)
- Round-base, Serial, Unrolled
Basics of Romulus

Two variants

- Nonce-based $N$-variants (NAE)
- Nonce Misuse-resistant $M$-variants (MRAE)
- Both consist of three members

Design goal: the best of lightweight AEAD built on TBC

- Small-state
- Rate 1 operation (number of input blocks per primitive call)
- Strong security
  - Both qualitatively and quantitatively
- Simple structure
### Family Members of Romulus

<table>
<thead>
<tr>
<th>Family</th>
<th>Name</th>
<th>$\tilde{E}$</th>
<th>$k$</th>
<th>$nl$</th>
<th>$n$</th>
<th>$t$</th>
<th>$d$</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romulus-N</td>
<td>Romulus-N1</td>
<td>Skinny-128-384</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>56</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>Romulus-N2</td>
<td>Skinny-128-384</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>96</td>
<td>48</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>Romulus-N3</td>
<td>Skinny-128-256</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>96</td>
<td>24</td>
<td>128</td>
</tr>
<tr>
<td>Romulus-M</td>
<td>Romulus-M1</td>
<td>Skinny-128-384</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>56</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>Romulus-M2</td>
<td>Skinny-128-384</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>96</td>
<td>48</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>Romulus-M3</td>
<td>Skinny-128-256</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>96</td>
<td>24</td>
<td>128</td>
</tr>
</tbody>
</table>

- $k$: key length, $nl$: nonce length, $t$: tweak main-block length
- $d$: counter length, $\tau$: tag length
- Skinny-x-y: Skinny with $x$-bit block, $y$-bit tweakey

N3 and M3 are most efficient, while not able to handle single input of $2^{50}$ bytes.
Romulus N-variants

- TBC $\widetilde{E}_K$ on tweak set $T = \{0, 1\}^t \times D \times B$ and message set $M = \{0, 1\}^n$
- State function $\rho : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n \times \{0, 1\}^n$
  - When AD is processed, the first output is ignored
- Based on iCOFB [CIMN16], with lots of changes/improvements
ρ function

Simple operation defined over bytes

- Byte matrix $G$
- Single-state (both red and blue lines can be independently computed)
- Partial input can be handle by truncation and padding
- Security condition for $\rho$: the same as COFB [CIMN16]
  - Unlike COFB, $G$ is applied to output side
  - Simplifies AD process (just XOR-chain)

Choice of $G$

- Modular form suitable to serial circuit, no need of MUX
- Small # of XOR, SW/HW-friendly

\[ G = \begin{pmatrix} G_s & 0 & 0 & \ldots & 0 \\ 0 & G_s & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & G_s & 0 \\ 0 & \ldots & 0 & 0 & G_s \end{pmatrix}, \quad G_s = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \]
Properties of Romulus-N

**Efficiency**
- Small state (TBC itself)
- Rate 1 \((n\text{-bit msg per call, } n + t\text{-bit AD per call})\)
- Small overhead for short message

**Security**
- \(n\text{-bit security with } n\text{-bit block TBC}\)
- Standard model: reduces to CPA security of TBC (TPRP)
  - Conservative, and no worry about the gap between the model and the instantiation
  - e.g. the use of weak permutation in Sponge constructions

**Limitations**
- Serial operation for both Enc/Dec
  - Reasonable for the applications of lightweight crypto
    - Parallel operation of many messages is always possible [BLT15]
    - Constraint devices are unlikely to process blocks in parallel for ASIC
Security Bounds for N-variants

\[ \text{Adv}_{\text{Romulus-N}}^{\text{priv}}(A) \leq \text{Adv}_{E}^{\text{tprp}}(A'), \]

\[ \text{Adv}_{\text{Romulus-N}}^{\text{auth}}(B) \leq \text{Adv}_{E}^{\text{tprp}}(B') + \frac{3q_d}{2^n} + \frac{2q_d}{2^\tau} \]

\( (q_d : \text{number of decryptions}, \tau : \text{tag length}) \)

**Previous**: AUTH contains \( O(\sigma_d/2^n) \) \( (\sigma_d : \text{total effective queried blocks in decryption}) \)

**Now**: essentially equal to \( \Theta \text{CB3} \) security, **no degradation in input length!**

... a quite unique security feature only achievable by TBC-based modes

**Proof**: similar technique as PFB [NS19]
Romulus M-variants

- (Fully) Nonce-misuse-resistance via SIV [RS06]
- Greatly shares Romulus-N components (easy to implement both)
- Proof: Use proof techniques of [NS19] and NaT MAC [CLS17]
Security Bounds for M-variants

Nonce-Respecting (NR) adversary:

\[
\text{Adv}_{\text{Romulus-M}}^{\text{priv}}(A) \leq \text{Adv}_{E}^{\text{tprp}}(A'),
\]

\[
\text{Adv}_{\text{Romulus-M}}^{\text{auth}}(B) \leq \text{Adv}_{E}^{\text{tprp}}(B') + \frac{5q_d}{2^n},
\]

Nonce-Misusing (NM) adversary w/ max \( r \) repetition of nonce in Enc:

\[
\text{Adv}_{\text{Romulus-M}}^{\text{nm-priv}}(A) \leq \text{Adv}_{E}^{\text{tprp}}(A') + \frac{4r\sigma_{\text{priv}}}{2^n},
\]

\[
\text{Adv}_{\text{Romulus-M}}^{\text{nm-auth}}(B) \leq \text{Adv}_{E}^{\text{tprp}}(B') + \frac{4rq_e + 5rq_d}{2^n}
\]

\((\sigma_{\text{priv}}: \text{total queried blocks in encryption})\)

Previous: AUTH includes \( O(\ell q_d/2^n) \), NM-AUTH includes \( O(r\ell q_d/2^n) \) & misses \( O(rq_e/2^n) \)

Now: no degradation in input length, except for nm-priv

... also very good security bounds, graceful security degradation for nonce repetition

* [CN19] subsequently informed us the need of incorporating the encryption queries and that they have proved a similar authenticity bound to ours.
Measuring the Efficiency of Romulus

Case of Romulus-N1 ($n = 128$):

**State**
- Skinny-128-384 has $n$-bit block + $3n$-bit tweakey
- State size = block ($n$) + effective part of tweak ($t = 1.5n$) + key ($k = n$) = $3.5n$
  - $t = 1.5n \rightarrow n$ for (AD/N) and $0.5n$ for (counter + domain bits)
  - Unused $0.5n$-bit tweakey does not need to be implemented (specific to Skinny)

**Rate (\# of input $n$-bit blocks per primitive call, for simplicity no AD)**
- 1 (for all N-variants)

**Security**
- $n$ bits

**Our efficiency measure (smaller is better) :** State/Rate = $3.5n$
## Detailed Comparison of NAE schemes \((n = k = 128)\)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Number of Primitive Calls</th>
<th>State Size</th>
<th>Rate</th>
<th>Security</th>
<th>Efficiency</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romulus-N1</td>
<td>(\lceil</td>
<td>A</td>
<td>- n/2n \rceil + \lceil</td>
<td>M</td>
<td>/ n \rceil + 1 )</td>
<td>3.5(n)</td>
</tr>
<tr>
<td>Romulus-N2</td>
<td>(\lceil</td>
<td>A</td>
<td>- n/1.75n \rceil + \lceil</td>
<td>M</td>
<td>/ n \rceil + 1 )</td>
<td>3.2(n)</td>
</tr>
<tr>
<td>Romulus-N3</td>
<td>(\lceil</td>
<td>A</td>
<td>- n/1.75n \rceil + \lceil</td>
<td>M</td>
<td>/ n \rceil + 1 )</td>
<td>3(n)</td>
</tr>
<tr>
<td>COFB</td>
<td>(\lceil</td>
<td>A</td>
<td>/ n \rceil + \lceil</td>
<td>M</td>
<td>/ n \rceil + 1 )</td>
<td>2.5(n)</td>
</tr>
<tr>
<td>(\Theta)CB3</td>
<td>(\lceil</td>
<td>A</td>
<td>/ n \rceil + \lceil</td>
<td>M</td>
<td>/ n \rceil + 1 )</td>
<td>4.5(n)</td>
</tr>
<tr>
<td>SpongeAE</td>
<td>(\lceil</td>
<td>A</td>
<td>/ n \rceil + \lceil</td>
<td>M</td>
<td>/ n \rceil + 1 )</td>
<td>3(n)</td>
</tr>
<tr>
<td>Beetle</td>
<td>(\lceil</td>
<td>A</td>
<td>/ n \rceil + \lceil</td>
<td>M</td>
<td>/ n \rceil + 2 )</td>
<td>2(n)</td>
</tr>
<tr>
<td>Ascon-128</td>
<td>(\lceil</td>
<td>A</td>
<td>/ 0.5n \rceil + \lceil</td>
<td>M</td>
<td>/ 0.5n \rceil + 1 )</td>
<td>3.5(n)</td>
</tr>
<tr>
<td>Ascon-128a</td>
<td>(\lceil</td>
<td>A</td>
<td>/ n \rceil + \lceil</td>
<td>M</td>
<td>/ n \rceil + 1 )</td>
<td>3.5(n)</td>
</tr>
</tbody>
</table>

- \(\Theta\)CB3: assuming \(n\)-bit nonce and \(n/2\)-bit counter
- SpongeAE: Duplex using \(3n\)-bit permutation with \(n\)-bit rate, \(2n\)-bit capacity.

**Romulus-N achieves the best efficiency among full \(n\)-bit secure schemes**
### Detailed Comparison of MRAE schemes \((n = k = 128)\)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Number of Primitive Calls</th>
<th>State Size ((S))</th>
<th>Rate ((R))</th>
<th>Security</th>
<th>Efficiency ((S/R))</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romulus-M1</td>
<td>([</td>
<td>A</td>
<td>+</td>
<td>M</td>
<td>n^{-n}}) + ([</td>
<td>M</td>
</tr>
<tr>
<td>Romulus-M2</td>
<td>([</td>
<td>A</td>
<td>+</td>
<td>M</td>
<td>1.75n^{-n}}) + ([</td>
<td>M</td>
</tr>
<tr>
<td>Romulus-M3</td>
<td>([</td>
<td>A</td>
<td>+</td>
<td>M</td>
<td>1.75n^{-n}}) + ([</td>
<td>M</td>
</tr>
<tr>
<td>SCT</td>
<td>([</td>
<td>A</td>
<td>+</td>
<td>M</td>
<td>n^{-}}) + ([</td>
<td>M</td>
</tr>
<tr>
<td>SUNDAE</td>
<td>([</td>
<td>A</td>
<td>+</td>
<td>M</td>
<td>n^{-}}) + ([</td>
<td>M</td>
</tr>
<tr>
<td>ZAE</td>
<td>([</td>
<td>A</td>
<td>+</td>
<td>M</td>
<td>2n^{-}}) + ([</td>
<td>M</td>
</tr>
</tbody>
</table>

**Romulus-M achieves the best efficiency among \(n \sim n/2\)-secure schemes**
## ASIC Implementations

**TSMC 65nm standard cell library (all synthesized by the same environment):**

<table>
<thead>
<tr>
<th>Variant</th>
<th>Cycles</th>
<th>Area (GE)</th>
<th>Minimum Delay (ns)</th>
<th>Throughput (Gbps)</th>
<th>Power (µW)</th>
<th>Energy (pJ)</th>
<th>Thput/Area (Gbps/kGE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romulus-N1 Low Area</td>
<td>1264</td>
<td>4498</td>
<td>0.8</td>
<td>0.1689</td>
<td>-</td>
<td>-</td>
<td>0.0376</td>
</tr>
<tr>
<td>Romulus-N1</td>
<td>60</td>
<td>6620</td>
<td>1</td>
<td>2.78</td>
<td>548</td>
<td>32.8</td>
<td>0.42</td>
</tr>
<tr>
<td>Romulus-N1 unrolled x4</td>
<td>18</td>
<td>10748</td>
<td>1</td>
<td>9.27</td>
<td>-</td>
<td>-</td>
<td>0.86</td>
</tr>
<tr>
<td>ACORN [ATHENA]</td>
<td>-</td>
<td>6580</td>
<td>0.9</td>
<td>8.8</td>
<td>-</td>
<td>-</td>
<td>1.36</td>
</tr>
<tr>
<td>Ascon Low Area [Official]</td>
<td>3078</td>
<td>4545</td>
<td>0.5</td>
<td>0.042</td>
<td>167</td>
<td>51402</td>
<td>0.01</td>
</tr>
<tr>
<td>Ascon Basic Iterative [Official]</td>
<td>6</td>
<td>8562</td>
<td>1</td>
<td>10.4</td>
<td>292.7</td>
<td>-</td>
<td>1.22</td>
</tr>
<tr>
<td>Ketje-Sr [ATHENA]</td>
<td>-</td>
<td>19230</td>
<td>0.9</td>
<td>1.11</td>
<td>-</td>
<td>-</td>
<td>0.06</td>
</tr>
</tbody>
</table>

- Power and Energy are estimated at 10 Mhz.
- Energy is for 1 TBC call

**Remarks:**

- Low-area Romulus-N1 is more efficient than low-area Ascon (one of the CAESAR winners)
- Ours are almost fully compliant to CAESAR API, Ascon implementations are custom API
## FPGA Implementations

### Xilinx Virtex 6 FPGA using ISE:

<table>
<thead>
<tr>
<th>Variant</th>
<th>Slices</th>
<th>LUTs</th>
<th>Registers</th>
<th>Max. Freq. (MHz)</th>
<th>Throughput (Mbps)</th>
<th>Throughput/Area (Mbps/Area)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romulus-N1</td>
<td>307</td>
<td>919</td>
<td>534</td>
<td>250</td>
<td>695</td>
<td>2.26</td>
</tr>
<tr>
<td>Romulus-N1 Unrolled ×4</td>
<td>597</td>
<td>1884</td>
<td>528</td>
<td>250</td>
<td>2300</td>
<td>3.85</td>
</tr>
<tr>
<td>Lilliput-I-128</td>
<td>391</td>
<td>1506</td>
<td>1017</td>
<td>185</td>
<td>657.8</td>
<td>1.68</td>
</tr>
<tr>
<td>Lilliput-II-128</td>
<td>309</td>
<td>1088</td>
<td>885</td>
<td>185</td>
<td>328.9</td>
<td>1.06</td>
</tr>
</tbody>
</table>

More schemes to be added for comparison
Some Implementation Details

- Utilize the fully linear tweakey scheduling, mostly routing and renaming bytes
  - Reverse tweakey schedule at the end of every TBC call, instead of keeping input
  - Very low area, only 67 XOR gates!
  - If we were to maintain tweakey state (due to modes/TBC), at least 320 FFs
- Lightweight core is suitable to full-unroll, excellent tread-off
  - Speeding up $\times 2$ by two-round unrolling: $\approx +1,000$ GEs, $+20\%$ of total area

Fig. Serial state update
Remus

IC-based Encryption (ICE)

- IC to TBC conversion, a variant of XHX [JLM+17]
  - Optimized to reduce state and computation for counter incrementation
- \((n(\text{block}), n(\text{key}))\)-BC can be used to implement \((n(\text{block}), 2n(\text{tweak}), n(\text{key}))\)-TBC
- Three versions, having different nonce-based mask derivation (\(L\) and \(V\))
Security Bounds of Remus and TGIF

- Remus bound = Romulus bound + ICE bound
  - for NR and NM adversaries
- ICE bound: $O(\sigma^2/2^c)$, $c = n$ for ICE 1 and 3, $c = 2n$ for ICE 2
- Updates on the bounds from the initial document, in a similar manner to Romulus
Concluding Remarks

Romulus: (what we believe) the best we can do for lightweight, highly reliable AEAD with TBC

- Very strong provable security bounds, in the standard model
  - N-variants: $n$-bit security equivalent to $\Theta CB3$
  - M-variants: $\approx n$-bit security as long as # of nonce repetition is small

- Skinny’s high security (CPA-security for single-key setting is enough)

- Rate 1 and minimum-state as TBC-based AE

Next Steps

- More HW implementations including M-variants
- MCU implementations
- Side-channel resistance
- (Third-party implementations are always welcome)
Concluding Remarks

Romulus: (what we believe) the best we can do for lightweight, highly reliable AEAD with TBC

- Very strong provable security bounds, in the standard model
  - N-variants: $n$-bit security equivalent to ΘCB3
  - M-variants: $\approx n$-bit security as long as # of nonce repetition is small

- Skinny’s high security (CPA-security for single-key setting is enough)

- Rate 1 and minimum-state as TBC-based AE

Next Steps

- More HW implementations including M-variants
- MCU implementations
- Side-channel resistance
- (Third-party implementations are always welcome)

Thanks!