Updates on Romulus, Remus and TGIF

Tetsu Iwata[1] Mustafa Khairallah[2]
Kazuhiko Minematsu[3] Thomas Peyrin[2]

Nagoya University[1] Nanyang Technological University[2] NEC[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)

(wikipedia)

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 (# of input blocks per primitive call)
- Strong security
  - Both **qualitatively** and **quantitatively**
- Simple structure

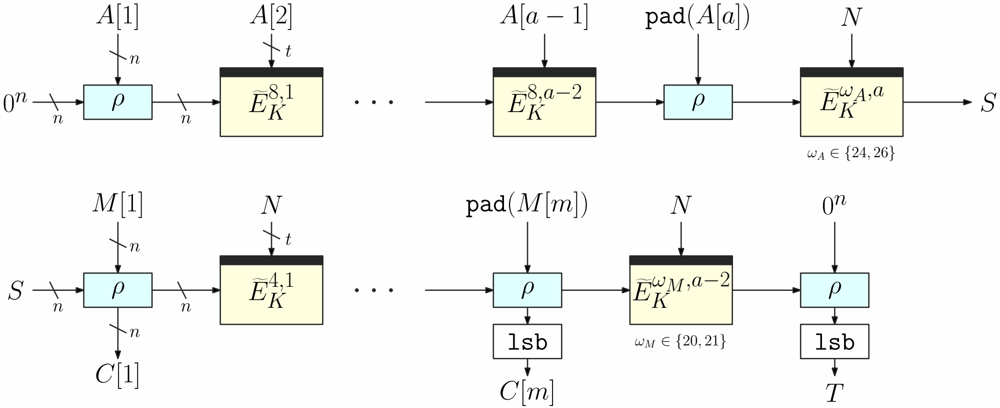
Family Members of Romulus

Family	Name	$\widetilde{E}$	k	nl	n	t	d	τ
Romulus-N	Romulus-N1	Skinny-128-384	128	128	128	128	56	128
	Romulus-N2	Skinny-128-384	128	96	128	96	48	128
	Romulus-N3	Skinny-128-256	128	96	128	96	24	128
Romulus-M	Romulus-M1	Skinny-128-384	128	128	128	128	56	128
	Romulus-M2	Skinny-128-384	128	96	128	96	48	128
	Romulus-M3	Skinny-128-256	128	96	128	96	24	128

- k : key length, nl : nonce length, t : tweak main-block length
- d : counter length, τ : 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

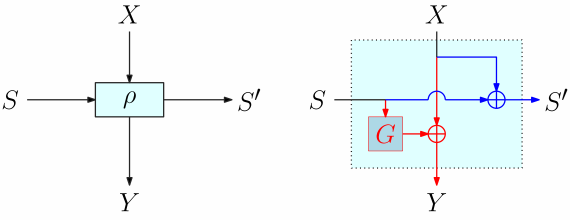


- TBC $\widetilde{E}_K$ on tweak set $\mathcal{T} = \{0,1\}^t \times \mathcal{D} \times \mathcal{B}$ and message set $\mathcal{M} = \{0,1\}^n$
- State function $\rho : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$
  - When AD is processed, the first output is ignored
- Based on iCOFB [CIMN16], 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 ρ : the same as COFB [CIMN16]
  - Unlike COFB, G is applied to output side
  - Simplifies AD process (just XOR-chain)



$$G = \begin{pmatrix} G_s & 0 & 0 & \dots & 0 \\ 0 & G_s & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & 0 & G_s & 0 \\ 0 & \dots & 0 & 0 & G_s \end{pmatrix}, \quad G_s = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Choice of G

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

Properties of Romulus-N

Efficiency

- Small state (TBC itself)
- Rate 1 (n-bit msg per call, $n + t$-bit AD per call)
- Small overhead for short message

Security

- n-bit security with n-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

$$\mathbf{Adv}^{\mathtt{priv}}_{\mathsf{Romulus\text{-}N}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathtt{tprp}}_{\widetilde{E}}(\mathcal{A}'),$$

$$\mathbf{Adv}^{\mathtt{auth}}_{\mathsf{Romulus\text{-}N}}(\mathcal{B}) \leq \mathbf{Adv}^{\mathtt{tprp}}_{\widetilde{E}}(\mathcal{B}') + \frac{3q_d}{2^n} + \frac{2q_d}{2^\tau}$$

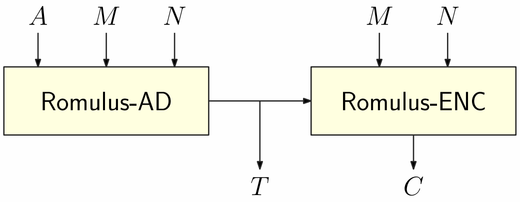
$$(q_d : \text{number of decryptions}, \tau : \text{tag length})$$

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

Now : essentially equal to ΘCB3 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 :

$$\mathbf{Adv}^{\mathtt{priv}}_{\text{Romulus-M}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathtt{tprp}}_{\widetilde{E}}(\mathcal{A}'),$$

$$\mathbf{Adv}^{\mathtt{auth}}_{\text{Romulus-M}}(\mathcal{B}) \leq \mathbf{Adv}^{\mathtt{tprp}}_{\widetilde{E}}(\mathcal{B}') + \frac{5q_d}{2^n}$$

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

$$\mathbf{Adv}^{\mathtt{nm\text{-}priv}}_{\text{Romulus-M}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathtt{tprp}}_{\widetilde{E}}(\mathcal{A}') + \frac{4r\sigma_{\text{priv}}}{2^n},$$

$$\mathbf{Adv}^{\mathtt{nm\text{-}auth}}_{\text{Romulus-M}}(\mathcal{B}) \leq \mathbf{Adv}^{\mathtt{tprp}}_{\widetilde{E}}(\mathcal{B}') + \frac{4rq_e + 5rq_d}{2^n}$$

(σ_{priv} : 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 ($n = k = 128$)

Scheme	Number of Primitive Calls	State Size (S)	Rate (R)	Security	Efficiency (S/R)	Inverse Free
Romulus-N1	$\lceil \frac{\lvert A\rvert - n}{2n} \rceil + \lceil \frac{\lvert M\rvert}{n} \rceil + 1$	$3.5n$	1	n	$3.5n$	Yes
Romulus-N2	$\lceil \frac{\lvert A\rvert - n}{1.75n} \rceil + \lceil \frac{\lvert M\rvert}{n} \rceil + 1$	$3.2n$	1	n	$3.2n$	Yes
Romulus-N3	$\lceil \frac{\lvert A\rvert - n}{1.75n} \rceil + \lceil \frac{\lvert M\rvert}{n} \rceil + 1$	$3n$	1	n	$3n$	Yes
COFB	$\lceil \frac{\lvert A\rvert}{n} \rceil + \lceil \frac{\lvert M\rvert}{n} \rceil + 1$	$2.5n$	1	$n/2 - \log n/2$	$2.5n$	Yes
ΘCB3	$\lceil \frac{\lvert A\rvert}{n} \rceil + \lceil \frac{\lvert M\rvert}{n} \rceil + 1$	$4.5n$	1	n	$4.5n$	No
SpongeAE	$\lceil \frac{\lvert A\rvert}{n} \rceil + \lceil \frac{\lvert M\rvert}{n} \rceil + 1$	$3n$	$1/3$	n	$9n$	Yes
Beetle	$\lceil \frac{\lvert A\rvert}{n} \rceil + \lceil \frac{\lvert M\rvert}{n} \rceil + 2$	$2n$	$1/2$	$n - \log n$	$4n$	Yes
Ascon-128	$\lceil \frac{\lvert A\rvert}{0.5n} \rceil + \lceil \frac{\lvert M\rvert}{0.5n} \rceil + 1$	$3.5n$	$1/5$	n	$17.5n$	Yes
Ascon-128a	$\lceil \frac{\lvert A\rvert}{n} \rceil + \lceil \frac{\lvert M\rvert}{n} \rceil + 1$	$3.5n$	$2/5$	n	$8.75n$	Yes

- ΘCB3: 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 ($n = k = 128$)

Scheme	Number of Primitive Calls	State Size (S)	Rate (R)	Security NR $\sim$ NM	Efficiency (S/R)	Inverse Free
Romulus-M1	$\lceil \frac{\|A\|+\|M\|-n}{2n} \rceil + \lceil \frac{\|M\|}{n} \rceil + 1$	$3.5n$	$2/3$	$n \sim n/2$	$5.25n$	Yes
Romulus-M2	$\lceil \frac{\|A\|+\|M\|-n}{1.75n} \rceil + \lceil \frac{\|M\|}{n} \rceil + 1$	$3.2n$	$7/11$	$n \sim n/2$	$5.03n$	Yes
Romulus-M3	$\lceil \frac{\|A\|+\|M\|-n}{1.75n} \rceil + \lceil \frac{\|M\|}{n} \rceil + 1$	$3n$	$7/11$	$n \sim n/2$	$4.71n$	Yes
SCT	$\lceil \frac{\|A\|+\|M\|}{n} \rceil + \lceil \frac{\|M\|}{n} \rceil + 1$	$4n$	$1/2$	$n \sim n/2$	$8n$	Yes
SUNDAE	$\lceil \frac{\|A\|+\|M\|}{n} \rceil + \lceil \frac{\|M\|}{n} \rceil + 1$	$2n$	$1/2$	$n/2$	$4n$	Yes
ZAE	$\lceil \frac{\|A\|+\|M\|}{2n} \rceil + \lceil \frac{\|M\|}{n} \rceil + 6$	$7n$	$2/3$	n	$10.5n$	Yes

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):

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

- 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 :

Variant	**Slices**	**LUTs**	**Registers**	**Max. Freq.** (MHz)	**Throughput** (Mbps)	**Throughput/Area** (Mbps/Area)
Romulus-N1	307	919	534	250	695	2.26
Romulus-N1 Unrolled $\times 4$	597	1884	528	250	2300	3.85
Lilliput-I-128	391	1506	1017	185	657.8	1.68
Lilliput-II-128	309	1088	885	185	328.9	1.06

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

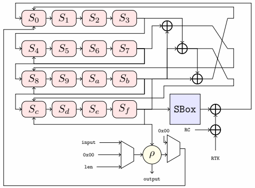
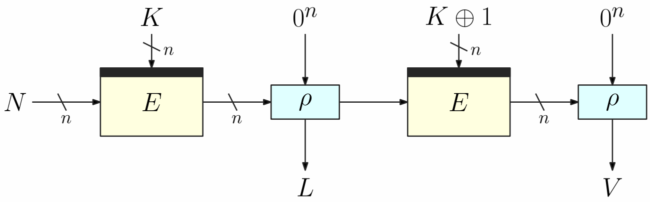


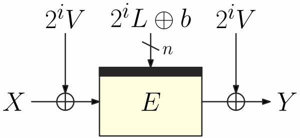
Fig. Serial state update

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 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)

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!