Updates on Romulus, Remus and TGIF

Tetsu Iwata[1], Mustafa Khairallah[2,3], Kazuhiko Minematsu[4], and Thomas Peyrin[2,3]

[1] Nagoya University, Japan
tetsu.iwata@nagoya-u.jp
[2] Nanyang Technological University, Singapore
[3] Temasek Laboratories @ NTU
mustafam001@e.ntu.edu.sg,thomas.peyrin@ntu.edu.sg
[4] NEC Corporation, Japan
k-minematsu@ah.jp.nec.com

Abstract. Romulus and Remus are two families of lightweight and efficient authenticated encryption with associated data (AEAD) proposed by the authors in 2019 as the submissions to NIST Lightweight Cryptography project. In this article, we report updates on the provable security and implementation results on Romulus and Remus since the initial submission documents.

Keywords: Romulus · Remus · authenticated encryption · lightweight cryptography · tweakable block cipher.

1 Introduction

Romulus and Remus are two families of authenticated encryption with associated data (AEAD) proposed in [10, 11]. They are candidates of NIST Lightweight Cryptography project (hereafter NIST LWC).

Romulus consists of a nonce-respecting variant called Romulus-N, and a misusing-resistant variant, called Romulus-M. Similarly, Remus consists of nonce-based Remus-N variant and misusing-resistant Remus-M variant. Each variant further consists of several parameters having different efficiency and security characteristics.

Both Romulus and Remus are based on Skinny, a tweakable block cipher (TBC) proposed at CRYPTO 2016 [2]. Romulus directly uses Skinny as a TBC, and the security proof is based on the standard pseudo-randomness assumption of the TBC. Remus is different and it uses Skinny as an ideal cipher to build another TBC. This allows even more efficiency from Romulus, at the cost of the security proof being based on the ideal-cipher model.

The underlying structure of Romulus and Remus is similar to COFB, a block cipher AE mode proposed at CHES 2017 [3], yet, they have numerous design improvements and optimizations as a TBC-based AEAD.

In this article, we report updates on the security and efficiency of Romulus and Remus. In more detail, we show the security bounds of Romulus and Remus can be improved with respect to the maximum message length. For example, nonce-based variant Romulus-N has security bound $O(q_d/2^\tau)$ plus the computational security term of Skinny, for q_d decryption queries with τ-bit tag. This is essentially identical to ΘCB3 (a well-known TBC-based AEAD) except for the constants, and the smallest possible as an AEAD algorithm of τ-bit tags. Other variants can be improved as well (though the level of improvement differs for each variant). We also present a fix of the security bound of Remus-N3 to include an additional term that does not change the overall bit security. The authenticity bounds of Romulus-M and Remus-M are also corrected and improved. The effect of the number of encryption queries is now incorporated, and the overall security bound is independent from the lengths of the queries for the information-theoretic case of Romulus-M.

We would like to note that since the variants of TGIF [12] share the same mode designs as Remus-N1, Remus-N2, Remus-M1 and Remus-M2, the provable security updates regarding these variants applies to TGIF, as well.

For update on implementation, we show the serialized hardware implementations that have sufficient speed and ultimately small gate size making our designs very competitive for extremely constrained environments. We also verify our theoretical performance/area estimates with comparison to real hardware implementations. We show different trade-offs between serialized, round-based and unrolled architectures for different design goals: low area, high throughput or high efficiency.

2 Update on Provable Security

We refer to the submission documents [10, 11] for the algorithms of the variants of Romulus and Remus, and the security notions (privacy and authenticity), and the accompanying notations. We improved the authenticity bounds. However, for completeness we also present the privacy bounds which did not change from the initial submission documents. For Remus-N3 we present fixed bounds for both privacy and authenticity.

2.1 Updated bounds of Romulus-N

For $A \in \{0,1\}^*$, we say A has a AD blocks if it is parsed as $(A[1], \ldots, A[a]) \xleftarrow{n,t} A$. Let $\tilde{a} = \lfloor a/2 \rfloor + 1$ which is a bound of actual number of primitive calls for AD. Similarly for plaintext $M \in \{0,1\}^*$, we say M has m message blocks if $|M|_n = m$. The same applies to ciphertext C. For encryption query (N, A, M) or decryption query (N, A, C, T) of a AD blocks and m message blocks, the number of total

TBC calls is at most $\tilde{a} + m$, which is called the number of *effective blocks* of a query.

While the specification assumes n-bit tag, we extend it to be (arbitrarily) fixed truncated to $\tau \in [\![n]\!]$ bits, and show the bounds with the case of τ-bit tag.

Theorem 1. *Let $\mathcal{A}$ be a (NR) privacy adversary against* Romulus-N *with time complexity t_A and with total number of effective blocks σ_{priv}. Moreover, let $\mathcal{B}$ be an NR authenticity adversary using q_d decryption queries, with total number of effective blocks for encryption and decryption queries σ_{auth}, and time complexity t_B. Then*

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

hold for some $(\sigma_{priv}, t_A + O(\sigma_{priv}))$-TPRP adversary $\mathcal{A}'$, and for some $(\sigma_{auth}, t_B + O(\sigma_{auth}))$-TPRP adversary $\mathcal{B}'$.

Theorem 1 holds for all the members of Romulus-N.

Improvements. Theorem 1 improves the previous authenticity bound of

$$\mathbf{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathcal{B}') + \frac{2\sigma_{\mathrm{dec}}}{2^n} + \frac{2q_d}{2^\tau},$$

where σ_{dec} denotes the total number of effective blocks in decryption queries. Privacy bound does not change. In particular, in the new authenticity bound, the information-theoretic term is independent of the input length and $O(q_d/2^\tau)$ is essentially minimum as an AEAD algorithm of τ-bit tag.

2.2 Updated bounds of Romulus-M

For an encryption query (N, A, M), we define the number of effective blocks as $\lfloor a/2 \rfloor + \lfloor m/2 \rfloor + 2 + m'$, where $(A[1], \ldots, A[a]) \xleftarrow{n,t} A$, $(M[1], \ldots, M[m]) \xleftarrow{n,t} M$ (or $(M[1], \ldots, M[m]) \xleftarrow{t,n} M$), and $(M[1], \ldots, M[m']) \xleftarrow{n} M$. For a decryption query (N, A, C, T), it is similarly defined by $(C[1], \ldots, C[m]) \xleftarrow{n,t} C$ or $(C[1], \ldots, C[m]) \xleftarrow{t,n} C$, and $(C[1], \ldots, C[m']) \xleftarrow{n} C$.

We first present bounds under nonce-respecting adversary:

Theorem 2. *Let $\mathcal{A}$ be a privacy adversary against* Romulus-M *that uses q_e encryption queries with time complexity t_A and with total number of effective blocks σ_{priv}. Let $\mathcal{B}$ be an authenticity adversary against* Romulus-M *using q_e encryption queries and q_d decryption queries, with total number of effective blocks*

for encryption and decryption queries σ_{auth} and with time complexity t_B. Then we have

$$\mathbf{Adv}^{\mathtt{priv}}_{\mathsf{Romulus\text{-}M}}(\mathcal{A}) \leqslant \mathbf{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathcal{A}'),$$
$$\mathbf{Adv}^{\mathtt{auth}}_{\mathsf{Romulus\text{-}M}}(\mathcal{B}) \leqslant \mathbf{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathcal{B}') + \frac{5q_d}{2^n}$$

for some $(\sigma_{priv}, t_A + O(\sigma_{priv}))$-TPRP adversary $\mathcal{A}'$, and $(\sigma_{auth}, t_B + O(\sigma_{auth}))$-TPRP adversary $\mathcal{B}'$.

We next present bounds under nonce-misusing adversary:

Theorem 3. *Let $\mathcal{A}$ be an NM-privacy adversary and let $\mathcal{B}$ be an NM-authenticity adversary, both are against* Romulus-M*, that use the same parameter as in Theorem 2, and can repeat a nonce at most $1 \leqslant r \leqslant 2^{n-1}$ times in encryption queries. Then we have*

$$\mathbf{Adv}^{\mathtt{nm\text{-}priv}}_{\mathsf{Romulus\text{-}M}}(\mathcal{A}) \leqslant \mathbf{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathcal{A}') + \frac{4r\sigma_{\mathrm{priv}}}{2^n},$$
$$\mathbf{Adv}^{\mathtt{nm\text{-}auth}}_{\mathsf{Romulus\text{-}M}}(\mathcal{B}) \leqslant \mathbf{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathcal{B}') + \frac{4rq_e + 5rq_d}{2^n}$$

for some $(\sigma_{priv}, t_A + O(\sigma_{priv}))$-TPRP adversary $\mathcal{A}'$ and $(\sigma_{auth}, t_B + O(\sigma_{auth}))$-TPRP adversary $\mathcal{B}'$.

Improvements and fix. These theorems show improvements of the authenticity bounds: previously, the information-theoretic terms of authenticity bounds were $2\ell q_d/2^n + 2q_d/2^n$ for nonce-respecting adversary and $2r\ell q_d/2^n + 2rq_d/2^n$ for nonce-misusing adversary, for maximum message block length ℓ.

The new bounds now correctly incorporate the number of encryption queries that were missing in the previous bounds, and we can say that, in the nonce-misusing case, if the total number of queries $(q_e + q_d)$ is sufficiently smaller than $2^n/r$, the scheme is secure. This does not change the claimed bit security. We note that subsequent to the update on our security bounds, Chakraborty and Nandi [5] informed us the need of incorporating the encryption queries and that they have proved a similar authenticity bound to ours.

2.3 Updated bounds of Remus-N

For $A \in \{0,1\}^*$, we say A has a AD blocks if $|A|_n = a$. Similarly for plaintext $M \in \{0,1\}^*$ we say M has m message blocks if $|M|_n = m$. The same holds for the ciphertext C. For encryption query (N, A, M) or decryption query (N, A, C, T) of a AD blocks and m message blocks, the number of total TBC calls is at most $a + m$, which is called the number of *effective blocks* of a query. As before $\tau \in [\![n]\!]$ is the tag length.

Theorem 4. *Let $\mathcal{A}$ be an NR privacy adversary against* Remus-N *using q_e encryption (construction) queries and q_p primitive queries with total number of effective blocks in encryption queries σ_{priv}. Moreover, let $\mathcal{B}$ be an NR authenticity adversary against* Remus-N *using q_e encryption queries and q_d decryption queries, with total number of effective blocks for encryption and decryption queries σ_{auth}, and q_p primitive queries. Then we have*

$$\begin{aligned}
\mathbf{Adv}^{\mathtt{priv}}_{\mathsf{Remus\text{-}N1}}(\mathcal{A}) &\leqslant \frac{9\sigma^2_{\mathrm{priv}} + 4q_p\sigma_{\mathrm{priv}}}{2^n} + \frac{2q_p}{2^n},\\
\mathbf{Adv}^{\mathtt{priv}}_{\mathsf{Remus\text{-}N2}}(\mathcal{A}) &\leqslant \frac{9\sigma^2_{\mathrm{priv}} + 4q_p\sigma_{\mathrm{priv}}}{2^{2n}} + \frac{2q_p}{2^n},\\
\mathbf{Adv}^{\mathtt{priv}}_{\mathsf{Remus\text{-}N3}}(\mathcal{A}) &\leqslant \frac{0.5\sigma^2_{\mathrm{priv}}}{2^{k-8}} + \frac{q_p\sigma_{\mathrm{priv}}}{2^k},\\
\mathbf{Adv}^{\mathtt{auth}}_{\mathsf{Remus\text{-}N1}}(\mathcal{B}) &\leqslant \frac{9\sigma^2_{\mathrm{auth}} + 4q_p\sigma_{\mathrm{auth}}}{2^n} + \frac{2q_p}{2^n} + \frac{3q_d}{2^n} + \frac{2q_d}{2^\tau},\\
\mathbf{Adv}^{\mathtt{auth}}_{\mathsf{Remus\text{-}N2}}(\mathcal{B}) &\leqslant \frac{9\sigma^2_{\mathrm{auth}} + 4q_p\sigma_{\mathrm{auth}}}{2^{2n}} + \frac{2q_p}{2^n} + \frac{3q_d}{2^n} + \frac{2q_d}{2^\tau},\\
\mathbf{Adv}^{\mathtt{auth}}_{\mathsf{Remus\text{-}N3}}(\mathcal{B}) &\leqslant \frac{0.5\sigma^2_{\mathrm{auth}}}{2^{k-8}} + \frac{q_p\sigma_{\mathrm{auth}}}{2^k} + \frac{3q_d}{2^n} + \frac{2q_d}{2^\tau}.
\end{aligned}$$

Improvements and fix. For all the variants of Remus-N, the previous authenticity bounds contain $2\ell q_d/2^n$, which has been changed to $3q_d/2^n$ in our new bounds. Unlike the case of Romulus-N, the authenticity bounds still implicitly depend on the message length via σ_{auth}. As a fix, the first authenticity term of Remus-N3 was added from the previous bound and the second term was slightly improved ($q_p\sigma_{\mathrm{auth}}/2^{k-8}$ to $q_p\sigma_{\mathrm{auth}}/2^k$) thanks to the revision of the proof. We comment that this fix does not change the total bit security of $(n-8)/2$ bits, where $k = n$.

2.4 Updated bounds of Remus-M

For encryption query (N, A, M) or decryption query (N, A, C, T) of a AD blocks and m message blocks, the number of total TBC calls is at most $a + 2m$, which is called the number of *effective blocks* of a query.

We first present bounds under nonce-respecting adversary:

Theorem 5. *Let $\mathcal{A}$ be an NR-privacy adversary against* Remus-M *using q_e (construction) encryption queries and q_p primitive queries with total number of effective blocks in encryption queries σ_{priv}. Moreover, let $\mathcal{B}$ be an NR-authenticity adversary using q_e encryption queries and q_d decryption queries, with total number of effective blocks for encryption and decryption queries σ_{auth}, and q_p primitive queries. Then we have*

$$\mathbf{Adv}^{\mathtt{priv}}_{\mathsf{Remus\text{-}M1}}(\mathcal{A}) \leqslant \frac{9\sigma^2_{\mathrm{priv}} + 4q_p\sigma_{\mathrm{priv}}}{2^n} + \frac{2q_p}{2^n},$$

$$\mathbf{Adv}^{\mathtt{priv}}_{\mathsf{Remus\text{-}M2}}(\mathcal{A}) \leqslant \frac{9\sigma^2_{\mathrm{priv}} + 4q_p\sigma_{\mathrm{priv}}}{2^{2n}} + \frac{2q_p}{2^n},$$
$$\mathbf{Adv}^{\mathtt{auth}}_{\mathsf{Remus\text{-}M1}}(\mathcal{B}) \leqslant \frac{9\sigma^2_{\mathrm{auth}} + 4q_p\sigma_{\mathrm{auth}}}{2^n} + \frac{2q_p}{2^n} + \frac{5q_d}{2^n},$$
$$\mathbf{Adv}^{\mathtt{auth}}_{\mathsf{Remus\text{-}M2}}(\mathcal{B}) \leqslant \frac{9\sigma^2_{\mathrm{auth}} + 4q_p\sigma_{\mathrm{auth}}}{2^{2n}} + \frac{2q_p}{2^n} + \frac{5q_d}{2^n}.$$

We next present bounds under nonce-misusing adversary:

Theorem 6. *Let $\mathcal{A}$ be an NM-privacy adversary and let $\mathcal{B}$ be an NM-authenticity adversary, both are against* Remus-M*, that use the parameters as specified in Theorem 5, and can repeat a nonce at most $1 \leqslant r \leqslant 2^{n-1}$ times in encryption queries. Then we have*

$$\mathbf{Adv}^{\mathtt{nm\text{-}priv}}_{\mathsf{Remus\text{-}M1}}(\mathcal{A}) \leqslant \frac{9\sigma^2_{\mathrm{priv}} + 4q_p\sigma_{\mathrm{priv}}}{2^n} + \frac{2q_p}{2^n} + \frac{4r\sigma_{\mathrm{priv}}}{2^n},$$
$$\mathbf{Adv}^{\mathtt{nm\text{-}priv}}_{\mathsf{Remus\text{-}M2}}(\mathcal{A}) \leqslant \frac{9\sigma^2_{\mathrm{priv}} + 4q_p\sigma_{\mathrm{priv}}}{2^{2n}} + \frac{2q_p}{2^n} + \frac{4r\sigma_{\mathrm{priv}}}{2^n},$$
$$\mathbf{Adv}^{\mathtt{nm\text{-}auth}}_{\mathsf{Remus\text{-}M1}}(\mathcal{B}) \leqslant \frac{9\sigma^2_{\mathrm{auth}} + 4q_p\sigma_{\mathrm{auth}}}{2^n} + \frac{2q_p}{2^n} + \frac{4rq_e}{2^n} + \frac{5rq_d}{2^n},$$
$$\mathbf{Adv}^{\mathtt{nm\text{-}auth}}_{\mathsf{Remus\text{-}M2}}(\mathcal{B}) \leqslant \frac{9\sigma^2_{\mathrm{auth}} + 4q_p\sigma_{\mathrm{auth}}}{2^{2n}} + \frac{2q_p}{2^n} + \frac{4rq_e}{2^n} + \frac{5rq_d}{2^n}.$$

Improvements and fix. Our improvements and fix for authenticity bounds can be described in the same manner to the case of Romulus-M.

2.5 Security proofs

We omit the proofs for the space limitation. For all schemes, the basic proof strategy has a similarity to the proofs of iCOFB [4], and the recent proposal of Naito and Sugawara's PFB mode [15], but we needed a careful case analysis. For analyzing M variants, we also employed a framework of MAC security proof proposed by Cogliati et al. [6]. The full proofs are presented in a separate paper [9].

3 Update on Implementation Results

3.1 ASIC efficiency

We have implemented several variants of the Romulus and Remus families against the TSMC 65nm standard cell library, in order to have a better understanding of the ASIC performance and cost and verify our estimations in Section 4. The implementations use the round-based Skinny implementation published on the

Skinny website [1]. The results in Table 1 show that Remus is very lightweight and efficient at the same time as it requires slightly above 3 KGE (3.5 KGE with a simple interface) for its round-based implementation for Remus-N1. Of course, even smaller (but slower) trade-off are possible by going with a serial implementation. At the other end of the spectrum in terms of security, the nonce-misuse resistant version Remus-M2 can be implemented in less than 5 KGE and provides 128-bit security, even in environments where randomness is not very reliable.

We have also implemented the serial, round-based and 4-round unrolled architectures of Romulus-N1. The figures are expected to be similar for Romulus-N2 and about 550 GEs smaller, and 12∼18% faster for Romulus-N3. Moreover, Remus and Romulus share a similar structure and our experimental results show that it is very cheap to convert the implementation to the misuse resistant variant. The implementation results of the Romulus-N1 and Remus-N1 4-round unrolled architecture show how Romulus and Remus take advantage of the versatility of Skinny for different architectures, where the throughput can be multiplied by 4 and the efficiency can be increased at a smaller area cost (about 75∼100%).

We have implemented Romulus-N1, Remus-N1 and Remus-N2 using the byte serial Skinny architecture and serialized feedback function for the modes. Serial implementations for different variants of Skinny have been proposed in [1, 2, 13]. Such implementations can be easily adapted for other Romulus and Remus members. We have implemented one of our smallest proposals, Remus-N1 using the byte-serial implementation of Skinny. Without the interface, Remus-N1 requires around 2100 GEs using the TSMC 65nm standard cell library including the Skinny logic, Remus-N1 Logic and both the key and state storage.

We compare our implementations to the ones of Ascon and ACORN, the winners of the lightweight portfolio of the CAESAR competition, and also Ketje-Sr. We have downloaded the implementations of ACORN and Ketje-Sr from the ATHENa benchmarking website [7] that were also used as part of the ASIC benchmarking efforts of the CAESAR Competition [14]. For Ascon, we have downloaded the serial and round based implementations from the designers website [8], since these implementations seem to have better performance metrics compared to the one on the ATHENa website, at the expense of using a different interface. We have synthesized all four implementations to the same technology we are using, TSMC 65nm, to allow fair comparisons. We show that Romulus achieves comparable performance (but Romulus security proofs are in the standard model) while Remus is very competitive in terms of low area. For example, all the variants of Remus can achieve throughput above 1 Gbps for less than 5000 GEs, while the performance of Ascon drops significantly when the area is reduced to this range. Additionally, we believe the different security models and goals should be taken into consideration when comparing the performance, as in Romulus and Remus we use a much stronger primitive without any round reduction.

Table 1: ASIC Implementations of Remus and Romulus using the TSMC 65nm standard cell library. Power and Energy are estimated at 10 Mhz. Energy is for 1 TBC call for Remus-N members and 2 TBC calls for Remus-M members.

Variant	Cycles	Area (GE)	Minimum Delay (ns)	Throughput (Gbps)	Power (μW)	Energy (pJ)	Thput/Area (Gbps/kGE)
Nonce-respecting schemes (64-bit data security)							
Remus-N1	44	3611	0.98	2.96	218.5	961.4	0.82
Remus-N1 Low Area	936	2834	0.8	0.1705	98	9172	0.06
Nonce-respecting schemes (128-bit data security)							
Remus-N2 Low Area	936	3700	0.8	0.1705	-	-	0.046
Remus-N2	44	4774	0.84	3.46	265.6	1168	0.72
Remus-N2 unrolled x4	14	9278	0.98	9.14	-	-	0.98
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 [7]	-	6580	0.9	8.8	-	-	1.36
Ascon Low Area [8]	3078	4545	0.5	0.042	167	51402	0.01
Ascon Basic Iterative [8]	6	8562	1	10.4	292.7	-	1.22
Ketje-Sr [7]	-	19230	0.9	1.11	-	-	0.06
Nonce-misuse resistant schemes (64-bit data security)							
Remus-M1	44(AD)/88(M)	3805	1.01	2.16	278.5	2446	0.56
Nonce-misuse resistant schemes (128-bit data security)							
Remus-M2	44(AD)/88(M)	4962	0.93	2.34	390.7	3440	0.47

3.2 FPGA efficiency

The FPGA results presented in Table 2 show that the FPGA implementations of Romulus and Remus follow the same trend as the ASIC implementations achieving between 156 and 285 Slices for the round based implementations and less than 500 Slices for the 4-round unrolled implementations. Table 2 shows that Romulus and Remus are significantly smaller, faster and more efficient than than the Deoxys-based design Lilliput for example.

4 Hardware Implementations

4.1 General Architecture and Hardware Estimates

The goal of the design of Romulus and Remus is to have a very small area overhead over the underlying TBC, specially for the round-based implementations. In order to achieve this goal, we set two requirements:

1. There should be no extra Flip-Flops over what is already required by the TBC, since Flip-Flops are very costly ($4 \sim 7$ GEs per Flip-Flop).
2. The number of possible inputs to each Flip-Flop and outputs of the circuits have to be minimized. This is in order to reduce the number of multiplexers required, which is usually one of the cause of efficiency reduction between the specification and implementation.

Table 2: FPGA Results for Remus and Romulus on the Xilinx Virtex 6 FPGA using ISE

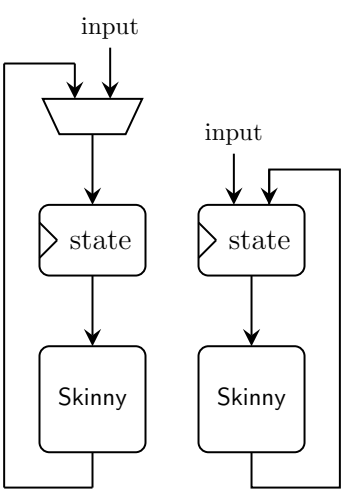
Variant	Slices	LUTs	Registers	Max. Freq. (MHz)	Throughput (Mbps)	Throughput/Area (Mbps/Area)
Nonce-respecting schemes (64-bit data security)						
Remus-N1	189	540	308	250	727.7	3.8
Remus-N1 †	550	1669	338	147.7	1358.84	2.4
Nonce-respecting schemes (128-bit data security)						
Remus-N2	225	757	358	250	727.7	3.23
Romulus-N1	307	919	534	250	695	2.26
Romulus-N1 †	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
Nonce-misuse resistant schemes (64-bit data security)						
Remus-M1	220	595	322	240	348	1.58
Nonce-misuse resistant schemes (128-bit data security)						
Remus-M2	279	816	397	219	317.6	1.13

† Unrolled x4.

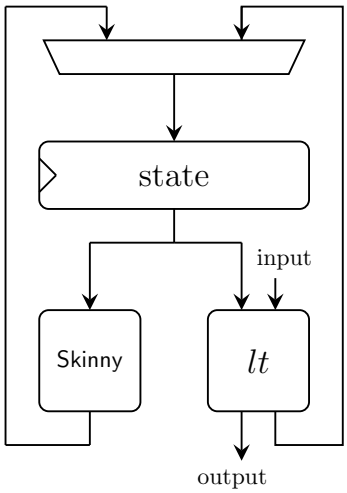
One of the advantages of Skinny as a lightweight TBC is that it has a very simple datapath, consisting of a simple state register followed by a low-area combinational circuit, where the same circuit is used for all the rounds, so the only multiplexer required is to select between the initial input for the first round and the round output afterwards (Figure 1(a)), and it has been shown that this multiplexer can even have lower cost than a normal multiplexer if it is combined with the Flip-Flops by using Scan-Flops (Figure 1(b)) [13]. However, when used inside an AEAD mode, challenges arise, such as how to store the key and nonce, as the key scheduling algorithm will change these values after each block encryption. The same goes for the block counter. In order to avoid duplicating the storage elements for these values; one set to be used to execute the TBC and one set to be used by the mode to maintain the current value, we studied the relation between the original and final value of the tweakey. Since the key scheduling algorithm of Skinny is fully linear and has very low area (most of the algorithm is just routing and renaming of different bytes), the full algorithm can be inverted using a very small circuit that costs 64 XOR gates for Romulus-N1 and 0 gates for Remus. Moreover, the LFSR computation required between blocks can be implemented on top of this circuit, costing $3 \sim 5$ extra XOR gates. This operation can be computed in parallel to ρ, such that when the state is updated for the next block, the tweakey key required is also ready. This costs only ~ 67 XOR gates as opposed to ~ 320 Flip-Flops that will, otherwise, be needed to maintain the tweakey value. Hence, the mode was designed with the architecture in Figure 1(b) in mind, where only a full-width state-register is used, carrying the TBC state and tweakey values, and every cycle, it is either kept without change, updated with the TBC round output (which includes a single round of the key scheduling algorithm) or the output of a simple linear transformation, which consists of ρ/ρ^{-1}, the unrolled inverse key schedule and the block counter.

Another possible optimization is to consider the fact that most of the area of Skinny comes from the storage elements, hence, we can speed up Romulus or Remus to almost double the speed by using a simple two-round unrolling, which costs $\sim 1,000$ GEs, as only the logic part of Skinny needs replication, which is only $< 20\%$ increase in terms of area for Romulus.

Romulus-M is estimated to have almost the same area as Romulus-N, except for an additional set of multiplexers in order to use the tag as an initial vector for the encryption part. This indicates that it can be a very lightweight choice for high security applications.



(a) Overview of the round based architecture of Skinny.



(b) Overview of the round based architecture of Remus. *lt*: The linear transformation that includes ρ, block counter and inverse key schedule.

Fig. 1: Expected architectures for Skinny and Remus

For the serial implementations we followed the currently popular bit-sliding framework [13] with minor tweaks. The state of Skinny is represented as the Feedback-Shift Register which typically operates on 8 bits at a time, while allowing the 32-bit MixColumns operation, given in Figure 2.

It can be viewed in Figure 2 that several careful design choices such as a lightweight serializable ρ function without the need of any extra storage and a lightweight padding/truncation scheme allow the low area implementations to use a very small number of multiplexers on top of the Skinny circuit for the state

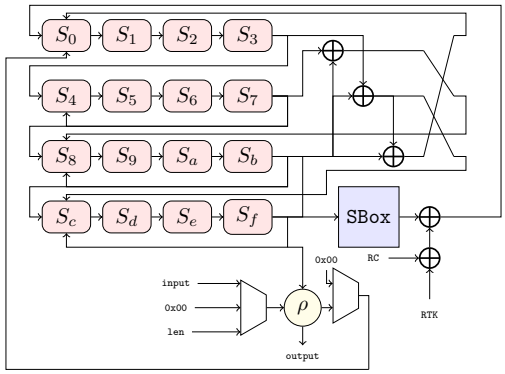


Fig. 2: Serial State Update Function Used in Romulus and Remus

update, three 8-bit multiplexer to be exact, two of which have a constant zero input, and $\sim$ 22 XORs for the ρ function and block counter. For the key update functions, we did several experiments on how to serialize the operations and we found the best trade-off is to design a parallel/serial register for every tweakey, where the key schedule and mode operations are done in the same manner of the round based implementation, while the AddRoundKey operation of Skinny is done serial as shown in Figure 2.

Usually there is a disparity between the theoretical estimate of the state size of a mode and the practical implementations. However, our practical implementations show that our theoretical estimates match exactly our implementations for both round based and serial implementations up to a constant term that covers the round constants of Skinny and the Finite State Machine of the API/interface, as shown in Table 3.

Table 3: Practical State Size of Romulus and Remus Variants

Variant	State Size
Romulus-N1	$448 + \mathcal{O}(1)$
Remus-N1	$256 + \mathcal{O}(1)$
Remus-N2	$384 + \mathcal{O}(1)$
Remus-M1	$256 + \mathcal{O}(1)$
Remus-M2	$384 + \mathcal{O}(1)$

4.2 Future Work

We have implemented some representative variants of Romulus and Remus and we have reported their area and performance. We plan to implement more variants,

do energy and power simulations and also provide side-channel and fault protected implementations.

Acknowledgments

The second and fourth authors are supported by Temasek Laboratories, Singapore.

References

1. The Skinny Cipher Website, `https://sites.google.com/site/skinnycipher/home`
2. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016)
3. Chakraborti, A., Iwata, T., Minematsu, K., Nandi, M.: Blockcipher-based authenticated encryption: How small can we go? In: CHES. Lecture Notes in Computer Science, vol. 10529, pp. 277–298. Springer (2017)
4. Chakraborti, A., Iwata, T., Minematsu, K., Nandi, M.: Blockcipher-based authenticated encryption: How small can we go? (full version of [3]). IACR Cryptology ePrint Archive **2017**, 649 (2017)
5. Chakraborty, B., Nandi, M.: Observation on Romulus-M. Private Communication (2019)
6. Cogliati, B., Lee, J., Seurin, Y.: New Constructions of MACs from (Tweakable) Block Ciphers. IACR Trans. Symmetric Cryptol. **2017**(2), 27–58 (2017)
7. George Mason University: ATHENa: Automated Tools for Hardware EvaluatioN. `https://cryptography.gmu.edu/athena/` (2017)
8. Github: ASCON-128 Hardware Design Document. `https://github.com/IAIK/ascon_hardware/blob/master/doc/ascon_hw_doc.pdf` (2017)
9. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Duel of the titans: The Romulus and Remus families of lightweight AEAD algorithms. IACR Cryptology ePrint Archive **2019**, 992 (2019)
10. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Remus v1. Submission to NIST Lightweight Cryptography Project (2019)
11. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Romulus v1. Submission to NIST Lightweight Cryptography Project (2019)
12. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T., Sasaki, Y., Sim, S.M., Sun, L.: TGIF v1. Submission to NIST Lightweight Cryptography Project (2019)
13. Jean, J., Moradi, A., Peyrin, T., Sasdrich, P.: Bit-sliding: A generic technique for bit-serial implementations of spn-based primitives. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 687–707. Springer (2017)
14. Kumar, S., Haj-Yihia, J., Khairallah, M., Chattopadhyay, A.: A comprehensive performance analysis of hardware implementations of caesar candidates. IACR Cryptology ePrint Archive **2017**, 1261 (2017)
15. Naito, Y., Sugawara, T.: Lightweight Authenticated Encryption Mode of Operation for Tweakable Block Ciphers. IACR Cryptology ePrint Archive **2019**, 339 (2019)