First-order masked ARMv7-M implementations of Romulus AEAD schemes

Alexandre Adomnicai

March 29, 2022

Abstract

This document briefly describes first-oder masked implementations of Romulus on ARMv7-M architectures which have been developped within the context of the NIST LWC standardization project.

1 Context

In 2018, the National Institute of Standards and Technology (NIST) initiated a process that started in 2018, with the goal of selecting the future Authenticated Encryption with Associated Data (AEAD) standard(s) for constrained environments [**?**]. AEAD algorithms ensure confidentiality, integrity, and authenticity of data in a single primitive. An important selection criterion, on top of security and performance, is the resilience against side-channel attacks since embedded devices are typical targets for such attacks. In order to assist the NIST in evaluating the LWC finalists in this regard, the Cryptographic Engineering Research Group from George Mason University issued a call for protected implementation of NIST LWC finalists[1]. The submissions have to follow a specific API so that it facilitates side-channel evaluations from the security labs involved in the process. The implementations described in this document were developped in this context and focus on Romulus [IKMP20][2], one of the 10 NIST LWC finalists, which is based on the Skinny-128 tweakable block cipher [BJK+16].

2 Implementation details

The first-order secure implementations presented in this document are based on a previous work employing an advanced bitslicing technique named *fixslicing* [AP20b, AP20a]. Therefore, all the code consists of bitwise operations only, which eases the integration of Boolean masking. Non-linear operations (i.e. `AND` and `OR` gates) are computed without additional randomness using the techniques detailed in Algorithms 1 and 2.

Algorithm 1: First-order Boolean masked `AND` gate without additional randomness from [BDCU17]

Input: (x_1, x_2) s.t. $x = x_1 \oplus x_2$; (y_1, y_2) s.t. $y = y_1 \oplus y_2$
Output: (z_1, z_2) s.t. $z = x \wedge y = z_1 \oplus z_2$
1 $z_1 = (x_1 \wedge y_1) \oplus (x_1 \vee \neg y_2)$
2 $z_2 = (x_2 \vee y_1) \oplus (x_2 \vee \neg y_2)$
3 **return** (z_1, z_2)

[1] https://cryptography.gmu.edu/athena/LWC/Call_for_Protected_Software_Implementations.pdf
[2] https://romulusae.github.io/romulus/

Algorithm 2: First-order Boolean masked `OR` gate without additional randomness from [BDCU17]

Input: (x_1, x_2) s.t. $x = x_1 \oplus x_2$; (y_1, y_2) s.t. $y = y_1 \oplus y_2$
Output: (z_1, z_2) s.t. $z = x \vee y = z_1 \oplus z_2$
1 $z_1 = (x_1 \wedge y_1) \oplus (x_1 \vee y_2)$
2 $z_2 = (x_2 \wedge y_1) \oplus (x_2 \wedge y_2)$
3 **return** (z_1, z_2)

Regarding Romulus-N and Romulus-M variants, the key is the only input which is split into 2 shares by the `generate_shares_encrypt` and `generate_shares_decrypt` functions. Since the tweakey schedule is fully linear, it is computed on both shares independently. At the Skinny-128-384+ level, the internal state is also split into 2 shares, where the initial shares are initialized to zero (i.e. the input block is not masked).

Regarding the Romulus-T variant, which uses a leakage-resilient mode of operation, the masking countermeasure is only applied to the key derivation and the tag generation functions since they constitute the only sensitive calculations that manipulate the secret key directly. Moreover, in addition to the key, the nonce is also split into 2 shares by the `generate_shares_encrypt` and `generate_shares_decrypt` functions. This choice is motivated by the fact that the nonce is used as input block for the key derivation function. Therefore, having a masked nonce allows to add randomness without considerable efforts.

Note that no hiding countermeasures have been integrated to these implementations so far.

3 Results of the preliminary security evaluation

No premilinary security evaluation has been undertaken.

References

[AP20a] Alexandre Adomnicai and Thomas Peyrin. Fixslicing - Application to Some NIST LWC Round 2 Candidates. NIST Lightweight Cryptography Workshop 2020, 2020.

[AP20b] Alexandre Adomnicai and Thomas Peyrin. Fixslicing AES-like Ciphers: New bitsliced AES speed records on ARM-Cortex M and RISC-V. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):402–425, Dec. 2020.

[BDCU17] Alex Biryukov, Daniel Dinu, Yann Le Corre, and Aleksei Udovenko. Optimal First-Order Boolean Masking for Embedded IoT Devices. In Thomas Eisenbarth and Yannick Teglia, editors, *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017*, volume 10728 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2017.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In *CRYPTO (2)*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the Titans: The Romulus and Remus Families of Lightweight AEAD Algorithms. *IACR Transactions on Symmetric Cryptology*, 2020(1):43–120, May 2020.