TinyJAMBU Update

Hongjun Wu and Tao Huang

Division of Mathematical Sciences
Nanyang Technological University
wuhongjun@gmail.com

18 September 2020

1 New Implementations

1.1 New Software Performance

At LWC 2019, Renner et al. compared the performance of the first round candidates on microcontrollers [1]. On the microcontroller STM32 F746ZG, TinyJAMBU-128 is the fastest in speed, the smallest in ROM usage, and the third smallest in RAM usage among the primary algorithms of the second round candidates. On the microcontroller Espressif ESP32, TinyJAMBU-128 is the fastest in speed, the smallest in ROM usage among the primary algorithms of the second round candidates.

Weatherley provided the following software performance data of TinyJAMBU. Weatherley's ranking on ESP32 [3] is different from that in [1]. Likely it is because that some implementations are optimized in [3].

Table 1: New Software Performance Data (*The speed is with respect to that of Chacha. **The ranking is among 32 primary algorithms of the second round candidates)

Platform	Average Speed*	Ranking**	Link
32-bit ARM Cortex M3	0.81	8th	[3]
32-bit ESP32	0.71	6th	[3]
8-bit AVR	2.13	8th	[4]

1.2 New Hardware Performance

For TinyJAMBU-128, the cipher core requires **1977 GE** when a key is loaded into the cipher core for every message (following the latest NIST LWC hardware

API) and eight rounds are implemented in parallel on UMC Technology 90nm ASIC. (In the second round submission, the TinyJAMBU-128 cipher core requires **1352 GE** when the key is a **fixed** value and eight rounds are implemented in parallel on UMC Technology 90nm ASIC.)

1.3 Protect against side channel attacks

Weatherley provided recently the preliminary masked C implementations of all the second round candidates [5]. The speed on TinyJAMBU-128 with 4-share mask is ranked 6th among the primitive algorithms of the second round candidates on the 32-bit ARM Cortex M3.

The round function of the keyed permutation of TinyJAMBU-128 is very simple, and it involves only four XOR and one NAND operations. It is thus relatively easy to implement TinyJAMBU against side channel attacks.

2 New Third-Party Analysis

In [2], Saha et al. analyzed the security margin of the nonce and associated data of TinyJAMBU against the forgery attack. In the TinyJAMBU design, our analysis shows that the differential forgery attack against nonce and associated data succeeds with probability probability at most 2^{-73}. In [2], it is shown that some NAND gates in the differential forgery attack are not independent, so the forgery attack against nonce and associated data succeeds with probability $2^{-70.64}$.

3 Platforms and metrics in which JAMBU performs better than current NIST standards

At LWC 2019, Yu and Aagaard provided the smallest AES cipher core on STMicro's 65nm ASIC which requires an area of 1960 GE [6].

For TinyJAMBU-128, the cipher core requires as small as **1352 GE** when the key is a fixed value and eight rounds are implemented in parallel on UMC Technology 90nm ASIC. It shows that TinyJABMU could be implemented much smaller than AES on ASIC.

For TinyJAMBU-128, the cipher core requires **1977 GE** when a key is loaded into the cipher core for every message (following the latest NIST LWC hardware API) and eight rounds are implemented in parallel on UMC Technology 90nm ASIC. It shows that TinyJAMBU cipher core is close to AES cipher core on ASIC (Note that the AES cipher core performs only encryption, while TinyJAMBU is an authenticated encryption algorhitm).

AES-GCM is not a lightweight authenticated encryption algorithm, and it requires 384 extra bits to store the state in addition to the AES implementation (128 bits to store the IV and counter, 128 bits to store the encryption of zero using AES, 128 bits to store the state of MAC). TinyJABMU should perform much better than AES-GCM for hardware lightweight cryptographic applications.

4 Target applications and use cases for which the candidate is optimized

TinyJAMBU is especially optimized when a key is already available in a lightweight cryptographic hardware device (for example, when there is a fixed secret key in the device). The TinyJAMBU-128 cipher core is only 1352 GE on ASIC for this type of application. For these lightweight devices, TinyJAMBU is significantly smaller in hardware than all the other second round candidates (the state size of TinyJAMBU is only 128 bits, and the round function of TinyJAMBU is very simple).

TinyJAMBU is optimized for the general hardware lightweight cryptographic applications. The TinyJAMBU-128 cipher core is only 1977 GE following the latest LWC hardware API.

TinyJAMBU is optimized for the use case of nonce misuse (defense in depth). When nonce is misused, the secret key in TinyJAMBU remains secure, and TinyJAMBU is still strong against forgery attack. (But TinyJAMBU does not provide strong encryption security when nonce is misused since the encryption security of TinyJAMBU is similar to that of Cipher Feedback mode.)

5 Planned Tweak Proposals

We plan to tweak the processing of nonce and associated data. In the current TinyJAMBU, the permutation P_{384} is used to process the 32-bit nonce blocks and 32-bit associated data blocks. In the planned tweak proposal, the permutation P_{640} will be used to replace P_{384}. There will be no change to the processing of plaintext blocks.

The reason of the tweak is to provide larger security margin for the protection of nonce and associated data.

References

[1] Sebastian Renner, Enrico Pozzobon, Jurgen Mottok. Benchmarking Software Implementations of 1st Round Candidates of the NIST LWC Project on Microcontrollers. NIST Lightweight Cryptography Workshop 2019. Available at `https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2019/documents/papers/benchmarking-software-implementations-lwc2019.pdf`

[2] Dhiman Saha, Yu Sasaki, Danping Shi, Ferdinand Sibleyras, Siwei Sun and Yingjie Zhang. On the Security Margin of TinyJAMBU with Refined

Differential and Linear Cryptanalysis. Available at `https://eprint.iacr.org/2020/1045`

[3] Rhys Weatherley. Lightweight Cryptography Primitives: Performance on 32-bit platforms. `https://rweather.github.io/lightweight-crypto/performance.html`

[4] Rhys Weatherley. Lightweight Cryptography Primitives: Performance on AVR. `https://rweather.github.io/lightweight-crypto/performance_avr.html`

[5] Rhys Weatherley. Lightweight Cryptography Primitives: Performance of Masked Algorithms. `https://rweather.github.io/lightweight-crypto/performance_masking.html`

[6] Jenny W. Yu and Mark D. Aagaard. Benchmarking and Optimizing AES for Lightweight Cryptography on ASICs. NIST Lightweight Cryptography Workshop 2019. Available at `https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2019/documents/papers/benchmarking-and-optimizing-aes-for-lwc-on-asics-lwc2019.pdf`