Update on Security Analysis and Implementations of KNOT

Wentao Zhang[1], Tianyou Ding[1], Bohan Yang[2], Zhenzhen Bao[3],
Zejun Xiang[4], Fulei Ji[1], Xuefeng Zhao[1], Chunning Zhou[1]

1.State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China;
2.Hardware Security and Cryptographic Processor Lab, IME, Tsinghua University;
3.Nanyang Technological University, Singapore;
4.Faculty of Mathematics and Statistics, Hubei Key Laboratory of Applied Mathematics, Hubei University;
{*zhangwentao, dingtianyou, jifulei, zhaoxuefeng, zhouchunning*}*@iie.ac.cn*
bohanyang@tsinghua.edu.cn
baozhenzhen10@gmail.com
xiangzejun@hubu.edu.cn

Abstract. KNOT is a family of permutation-based and bit-sliced lightweight AEAD and hashing algorithms. In this report, we will review the new cryptanalytic results of KNOT and summarize the new implementation results of KNOT in both software and hardware.

1 Security Analysis

In the design document[15], we have evaluated in detail the security of the underlying KNOT permutations against differential, linear, division-based integral, impossible differential and algebraic cryptanalysis. Although distinguishers of the permutations can give insights in the resistance of the AEAD and hash primitives against various cryptanalytic attacks, they usually can not be directly used in an attack. To have a better understanding of the security of KNOT, we furthermore studied the best differential and linear distinguishers with constraints in [6], which are corresponding to 7 important attack models and can be directly used to mount attacks on KNOT-AEAD or KNOT-Hash.

In [6], for security analysis of KNOT-AEAD, we considered 5 attack models, which are differential distinguishing attack targeting the initialization phase, linear key-recovery attack targeting the initialization phase, linear distinguishing attack targeting the initialization phase, linear distinguishing attack targeting the encryption phase and forgery attack targeting the finalization phase. For security analysis of KNOT-Hash, we considered 2 attack models, one is collision attack, the other is near-collision attack.

For the security margin of KNOT-AEAD, we consider this problem from 2 aspects:

1. **Scenario I**: the amount of data under an adversary's control is limited to the data limit M. A typical scenario is that the adversary can only obtain data under a single key (the concrete values of M can be found in Table 1). For the primary member of KNOT-AEAD, the data limit is 2^{64} data blocks, which is sufficient for lightweight applications in practice. Under this scenario, the adversary can either recover the secret key or distinguish the key-stream blocks from binary pseudorandom sequences.
2. **Scenario II**: the amount of data under an adversary's control is limited to 2^s, where s is the security bound (the concrete values of s can be found in Table 1). Which means, the computational complexity of an adversary is upper limited to 2^s executions of the underlying permutation. Under this scenario, an adversary can obtain data under multiple keys; with multiple-key data complexity, the adversary can only mount distinguishing attacks.

The KNOT family has three different state sizes: 256, 384 and 512 bits. Especially, the primary KNOT-AEAD member and the primary KNOT-Hash member both have a state of 256 bits. The KNOT-AEAD family has 4 members. Let KNOT-AEAD(k, b, r) denote a KNOT-AEAD member with k-bit key, b-bit state and r-bit rate. Based on the results in [6], Table 1 presents the rounds of the best distinguishers for each KNOT-AEAD under Scenario I and Scenario II respectively. The KNOT hash family also has 4 members. For KNOT-Hash, a similar categorization of attack scenarios is followed: Scenario I with M data limitation, Scenario II with 2^s data limitation. Let KNOT-Hash(n, b, r, r') denote a KNOT-Hash member with n-bit hash output, b-bit state, r-bit absorbing rate and r'-bit squeezing rate. Based on the results in [6], Table 2 presents the rounds of the best distinguishers for collision attacks on each KNOT-Hash under Scenario I and Scenario II respectively.

The results in Table 1 show that, under Scenario I, each KNOT-AEAD have a quite generous security margin against the 4 attack models considered, as the rounds of the best distinguisher is less than 50% of the full number of rounds respectively for the initialization, encryption and finalization phase. However, under

Table 1. Best Known Security Analysis of KNOT-AEAD Family

Name	Rounds			Data Limit M	Rounds-Scen.I			Security Bound s	Rounds-Scen.II		
	nr_0	nr	nr_f		Init.	Enc.	Final.		Init.	Enc.	Final.
KNOT-AEAD(128, 256, 64)	52	28	32	2^{64}	14	11	13	125	27	24	26
KNOT-AEAD(128, 384, 192)	76	28	32	2^{64}	13	12	14	128	27	25	27
KNOT-AEAD(192, 384, 96)	76	40	44	2^{96}	21	18	20	189	40	36	39
KNOT-AEAD(256, 512, 128)	100	52	56	2^{128}	27	24	26	253	53	49	53

1. KNOT-AEAD(128, 256, 64) is the primary AEAD member.
2. nr_0, nr and nr_f denote the number of rounds for the initialization (Init.), the processing of plaintext (Enc.) and the finalization (Final.) respectively.
3. data limit: the number of input data blocks under one key.
4. Rounds-Scen.I/Rounds-Scen.II: rounds of the best distinguishers under Scenario I/Scenario II, for the Init., Enc. and Final. phase respectively.
5. security bound: the logarithm based 2 of the attack cost, the unit is the underlying KNOT permutation.

Scenario II, taking multiple-key attacks into account, the results in Table 1 suggest that the values of nr and nr_f need to be increased to have a more comfortable security margin.

From Table 2, even in Scenario II, all KNOT-Hash members have sufficient security margin against the 2 attack models considered.

Table 2. Best Known Security Analysis of KNOT-Hash Family

Name	Rounds	Data Limit M	Rounds-Scen.I	Securiy Bound s	Rounds-Scen.II
	nr_h				
KNOT-Hash(256,256,32,128)	68	2^{64}	13	112	23
KNOT-Hash(256,384,128,128)	80	2^{64}	13	128	27
KNOT-Hash(384,384,48,192)	104	2^{96}	20	168	35
KNOT-Hash(512,512,64,256)	140	2^{128}	26	224	47

1. where KNOT-Hash(256, 256, 32, 128) is the primary hash member.
2. data limit: the message length limit.
3. Rounds-Scen.I/Rounds-Scen.II: rounds of the best distinguishers for collision attacks under Scenario I/Scenario II.
4. security bound: the logarithm based 2 of a collision attack cost, the unit is the underlying KNOT permutation.

2 Software and Hardware Implementations

2.1 Software Implementations on 64-bit Platform

From the measurement results of the toolkit SUPERCOP [13] by third-parties, we chose the results of KNOT on an AMD computer and that on an Intel computer to present in Table 3; speeds of AES-GCM on the two computers are also presented (note that the performance of AES can be supreme when using AES-NI that is available on the two computers). From Table 3, performance of all members of KNOT on 64-bit platforms are satisfactory.

2.2 Software Implementations on Microcontrollers

All members of KNOT show remarkable characteristics and friendliness on microcontrollers. Since KNOT-AEAD is inverse-free, only a small overhead is needed for supporting authenticated decryption on top of authenticated encryption. Furthermore, supporting hashing on top of AEAD costs limited additional resources.

On 8-bit Platform. We developed two sets of implementations of KNOT on 8-bit MCUs. One set is targeted at minimizing ROM requirement, and the other is targeted at optimizing speed. In both sets, the implementation of the permutation is in the bit-sliced way. They are performed in a constant time regardless values of input data. The cores of our implementations are written in assembly with C APIs that compliant with the SUPERCOP API. These two sets of implementations were submitted to the benchmarking project [10]. The resulted performance of KNOT family on 8-bit MCUs are collected in Table 4.

We also measured our speed-priority implementations of all members of KNOT independently. From our measurements, all members of KNOT-AEAD can be implemented with code size less than 2500 bytes, and all

Table 3. The speed of KNOT-AEAD ENC and KNOT-HASH on SUPERCOP

	AMD EPYC			Intel Xeon Gold		
LEN (adlen = mlen = msglen)	64	1536	long	64	1536	long
KNOT-AEAD(128,256,64)	29.69	23.38	23.09	29.33	24.98	25.35
KNOT-AEAD(128,384,192)	27.19	16.07	15.44	32.17	17.30	16.66
KNOT-AEAD(192,384,96)	55.16	41.87	41.18	61.45	46.85	46.07
KNOT-AEAD(256,512,128)	61.41	41.60	40.72	66.17	44.55	43.59
KNOT-Hash(256,256,32,128)	126.88	112.25	111.6	132.03	117.17	115.97
KNOT-Hash(256,384,128128)	95.31	64.79	63.38	104.50	70.67	69.08
KNOT-Hash(384,384,48,192)	254.69	227.29	224.49	275.84	246.37	244.01
KNOT-Hash(512,512,64,256)	270.62	216.72	213.90	289.81	233.88	231.29
AES128GCMV1	5.16	0.94	0.82	6.27	0.94	0.79
AES256GCMV1	12.19	1.15	0.66	12.16	1.09	0.64

[1] The presented results are from `https://bench.cr.yp.to/results-aead.html` and `https://bench.cr.yp.to/results-hash.html`.
[2] LEN: Data length, is in Bytes; speed is in cycle per byte.
[3] Computer information : 2019 AMD EPYC 7702;amd64;Zen2 (830f10); 64 x 2000MHz; genji346, supercop-20191017.
[4] Computer information : 2019 Intel Xeon Gold 6248; amd64; CascadeLake (50657); 20 x 2500MHz; pmnod076, supercop-20191017.
[5] For KNOT-AEAD, the speed of authenticated decryption is almost equal to that of the authenticated encryption (which is not generally hold by deffirent designs).

members of KNOT-Hash can be less than 1600 bytes. To support full functionality (authenticated encryption, authenticated decryption, and hashing), all KNOT-Pairs require less than 3000 bytes of ROM, less than 120 bytes of RAM. Specifically, for the primary pair KNOT-Pair I, AEAD requires less than 2000 bytes of ROM, 70 bytes of RAM, and runs (executing both authenticated enc. and dec.) at an average speed faster than 2500 cycles per byte; hashing requires less than 1000 bytes of ROM, 40 bytes of RAM, and runs at an average speed faster than 4500 cycles per byte. Supporting both AEAD and hashing requires less than 2500 bytes of ROM.

On 32-bit Platforms. The implementations of KNOT family on 32-bit platforms can apply the technique of bit-interleaving [5]. Adopting bit-interleaving and in the bit-sliced way, we implemented all members of KNOT for 32-bit MCUs. The implementations are all written in inline assembly with C code. The C APIs are compliant with the SUPERCOP API, so that we can submitted our implementations to the benchmarking project [10]. Some of the resulted performances of KNOT family on 32-bit MCUs are collected in Table 4.

Performance benchmarks of KNOT-AEAD on lwc.las3.de. The platform introduced in [10] provides benchmarks of software implementations of AEAD of the second-round candidates. This platform facilitates comparisons among different algorithms on different microcontrollers (MCUs). To generate benchmarks of KNOT on this platform, we submitted several sets of our implementations of KNOT for 8-bit and 32-bit MCUs. Table 4 presents performance benchmarks of members of KNOT-AEAD provided by this platform, where two sets of implementations are included. One set is targeted at optimizing speed (fastest), and the other is targeted at minimizing ROM (smallest). The performance of AES128K96N provided by this platform is also presented.

From the results on Arduino Uno R3 (8-bit MCU), members of KNOT-AEAD, especially the primary KNOT-AEAD member, achieve the lowest ROM and outstanding speed. For other MCUs, the primary AEAD member performs better than AES128K96N both in terms of speed and ROM on all the four 32-bit microcontrollers[1]. On all MCUs, the most significant advantage of members of KNOT-AEAD is the low ROM requirement.

2.3 Hardware Implementations of KNOT

We prepared two different sets of KNOT implementations to evaluate the intrinsic hardware performance and the flexibility of KNOT. The first is the basic iterative implementations for all four versions of KNOT-AEAD and KNOT-HASH. We also extended our KNOT-AEAD implementations to be compliant with the LWC Hardware API [9]. The basic iterative implementation results are compared to the existing hardware implementations of AES-GCM from NIST SP800-38D [7].

The basic iterative implementations have full-sized I/O interfaces for plaintext, ciphertext, etc. We note that architectures with full-sized interface are not suitable to be directly used as the top-module of implementations on FPGAs and ASICs due to potential oversized I/O utilization. However, they can function as independent IP cores for SoCs or reference implementations for optimizations.

To evaluate the hardware performance of our basic iterative implementations, the proposed design was synthesized with Synopsys Design Compiler G-2012.06-SP5 to the NANGATE45 open cell library(PDKv1 3 v2010 12). With the wire load model specified as 5K_hvratio_1_1, all our implementations could pass a frequency constraint of 800MHz.

[1] On the plateform, there is no available benchmarks of AES128K96N on Arduino Uno R3.

Table 4. Performance benchmarks of KNOT-AEAD on microcontrollers from the platform lwc.las3.de

		Arduino Uno R3		STM32F1 "bluepill"		Espressif ESP32		STM32 NUCLEOF746ZG			Sipeed Maixduino	
	Cipher	avg time	ROM	avg time	ROM	avg time	ROM	avg time	ROM	RAM	avg time	ROM
fastest	KNOT-AEAD(128,256,64)	2362.62	1384	176.714	6464	53.122	8544	26.587	6868	558	14.35	2240
	KNOT-AEAD(128,384,192)	3148.28	1682	270.33	22540	132.01	3728	36.15	2188	526	20.97	3264
	KNOT-AEAD(192,384,96)	4802.83	1668	391.89	17388	200.47	3504	55.21	2752	542	31.76	3136
	KNOT-AEAD(256,512,128)	7717.22	1884	700.23	3400	360.87	4064	100.51	3392	582	32.24	3904
smallest	KNOT-AEAD(128,256,64)	5432.69	1142	278.291	1264	69.405	2224	27.719	1856	570	14.35	2240
	KNOT-AEAD(128,384,192)	5959.75	1220	289.45	2296	132.01	3728	36.15	2188	526	20.97	3264
	KNOT-AEAD(192,384,96)	9132.62	1206	530.22	2756	200.47	3504	77.43	2744	670	31.76	3136
	KNOT-AEAD(256,512,128)	13811.2	1276	700.23	3400	360.87	4064	100.51	3392	582	32.24	3904
	AES128K96N	-	-	337.2	9908	67.75	14832	36.22	9836	1185	24.11	14272

[1] The presented results are from https://lwc.las3.de/table.php.
[2] RAM is in Bytes, ROM is in Bytes,Time is microsecond.
[3] The top half is for implementations targeted at optimizing speed, the second half is for that targeted at minimizing ROM.
[4] To reflect the real ROM requirement, the ROM required for nocrypt.memcpy is subtracted from what was presented in the webpage.

Table 5. Characteristics of KNOT and related implementations.

Designs	Area (GE)	Frequency (MHz)	Throughput (Mbps)	Throughput/Area (Mbps/GE)	Power (mW)	Energy (pJ/byte)
KNOT-AEAD(128,256, 64)	4553	800	1828.57	0.40	5.16	22.57
KNOT-AEAD(128,384,192)	7482	800	5485.7	0.73	8.2	11.95
KNOT-AEAD(192,384, 96)	6709	800	1920	0.29	7.58	31.58
KNOT-AEAD(256,512,128)	8849	800	1969.23	0.22	10.1	41.03
KNOT-HASH(256,256, 32,128)	3803	800	376	0.10	4.17	88.72
KNOT-HASH(256,384,128,129)	5850	800	1280	0.22	6.38	39.88
KNOT-HASH(384,384, 48,192)	5608	800	369	0.07	6.22	134.85
KNOT-HASH(512,512, 64,256)	7420	800	365	0.05	8.24	180.6
AES-GCM @ 40nm TSMC [2]	14993	500	1455	0.10	N.A.	N.A.
AES-GCM @ 130nm [11]	34500	200	2560	0.07	N.A.	N.A.

The results of our implementations are collected in Table 5. For these round-based implementations, the KNOT-AEAD(128, 256, 64) and the KNOT-HASH(256, 256, 32, 128) from the KNOT-Pair I have the smallest area of 4553 GE and 3803 GE respectively. The KNOT-AEAD(128, 384, 192) and KNOT-HASH(256, 384, 128, 129) from the KNOT-Pair II have the largest throughput and the best energy efficiency while processing long messages.

Compared to existing AES-GCM implementations, all our KNOT-AEAD implementations have smaller area and better throughput. The implementation area of primary version KNOT-AEAD(128, 256, 64) is 3× smaller than [2] under a similar technology node. The implementation of KNOT-AEAD(128, 384, 192) has a largest throughput of 5.486Gbps, which is 2.77× faster than [2]. For a fairer comparison, a metric *TOA* defined as Throughput/Area is used. The implementations of KNOT-AEAD(128, 256, 63) and KNOT-AEAD(128, 384, 192) are 4× and 6.3× better than [2] in TOA, respectively.

We extended our hardware implementation to be fully compliant with the LWC Hardware API. The KNOT-LWC implementations was delivered to the CERG-GMU LWC Team for further benchmarking and a better comparison to other candidates. The LWC Hardware API also provides a common and flexible interface with popular existing micro-controllers.

2.4 Capability of Integrating Side-Channel Countermeasures

For application scenarios where side-channel resistance is critical, KNOT by design can be implemented efficiently. Implementations of the KNOT families could follow the bit-slice style without using the look-up tables, which helps to mitigate the threat of cache-timing attack. The 4-bit S-box used in KNOT comes from the same family as popular block ciphers, such as RECTANGLE and PRESENT. The existence of countermeasures for S-boxes of RECTANGLE and PRESENT implies the feasibility of implementing efficient first-order and higher-order masking, or threshold implementations for KNOT families. According to the analysis on stream ciphers in [8], in the scenario of DPA attacks, the side-channel protection of KNOT should focus on the initialization phase, while the protection of Ascon need to take care both the initialization and finalization [1]. It enables efficient TI or masking implementations of KNOT.

3 Target Applications and Use Cases of KNOT

Based on the implementation results of KNOT in section 2, owing to the bit-slice style, KNOT allows for very efficient and flexible implementations in both hardware and software environments. Due to its Duplex/Sponge modes, compact state size, 4-bit Sbox and a bit permutation based diffusion layer, KNOT is well-suited for different constrained devices. Moreover, the implementation of the round function can be reused in the KNOT-AEAD and KNOT-Hash of the same KNOT-Pair, which reduces the hardware area or software ROM. The bit-slice style, together with carefully selected S-box, enables efficient side-channel resistant implementations of KNOT.

4 Planned Tweak Proposal

We will consider the following 2 tweaks:

1. If the security against multi-key attacks is essential for lightweight AEADs, we will increase the number of rounds nr and nr_f of KNOT-AEADs. Take the primary member of KNOT-AEAD for example, we will increase nr from 28 to 36 and increase nr_f from 32 to 40. As a result, the security margin of the primary KNOT-AEAD w.r.t the number of unattacked rounds is approximately 33% against an adversary with 2^{125} data complexity and 2^{125} time complexity; the software speed (or the hardware throughput) is decreased approximately 28% with the ROM and RAM requirements (or hardware area) almost unchanged.
2. To support hashing on top of AEAD more seamlesssly, within each KNOT-Pairs, the LFSRs used to generate round constants of the underlying permutation is planned to be defined as exactly the same.

5 Other Information

More details of the security analysis and implementations of KNOT are available at the KNOT website [14].

References

1. Adomnicai, A., Fournier, J. J., Masson, L. Masking the Lightweight Authenticated Ciphers ACORN and Ascon in Software. IACR Cryptol. ePrint Arch., `https://eprint.iacr.org/2018/708`.
2. AES-GCM Authenticated Encrypt/Decrypt Engine,CAST Inc., `https://www.cast-inc.com/security/encryption-primitives/aes-gcm/`, accessed September 17, 2020.
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponge functions, 2011. `https://keccak.team/files/SpongeFunctions.pdf`.
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: Single-pass authenticated encryption and other appli- cations, Selected Areas in Cryptography - SAC 2011, Revised Selected Papers, LNCS, vol. 7118, 320C337, Springer, 2011.
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G. Keccak implementation overview. `http://keccak.noekeon.org/Keccakimplementation-3.1.pdf`, 2012.
6. Ding T., Zhang W., Ji F., Zhou C., An Automatic Search Tool for Iterative Trails and its Application to KNOT, PRESENT, GIFT-64 and RECTANGLE, soon available at `https://eprint.iacr.org/` (before 22th September, 2020).
7. Dworkin, Morris J., Sp 800-38d. recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac, National Institute of Standards & Technology, 2007.
8. Gierlichs, B., Batina, L., Clavier, C., Eisenbarth, T., Gouget, A., Handschuh, H., et al. Susceptibility of eSTREAM candidates towards side channel analysis, 2008.
9. Kaps, J. P., Diehl, W., Tempelmeier, M., Homsirikamol, E., Gaj, K. Hardware API for Lightweight Cryptography. `https://cryptography.gmu.edu/athena/index.php`.
10. Renner S., Pozzobon E., Mottok J., OTH Regensburg initiative, `https://lwc.las3.de/`.
11. Satoh, A., Sugawara, T., Aoki, T., High-Performance Hardware Architectures for Galois Counter Mode, IEEE Transactions on Computers, vol 28, No.7, 917-930, 2009.
12. Sovyn Y., Khoma V., Podpora M. Comparison of three cpu-core families for iot applications in terms of security and performance of AES-GCM. *IEEE Internet Things J.*, 7(1):339–348, 2020.
13. SUPERCOP, `https://bench.cr.yp.to/supercop.html`.
14. The KNOT website, `http://www.sklois.ac.cn/kycg1/dbcg/202006/t20200617_565008.html`, 2020.
15. Zhang, W., Ding, T., Yang, B., Bao. Z., Xiang Z., Ji F., Zhao X., KNOT - Submission to Round 2 of the NIST Lightweight Cryptography Standardization process, `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/knot-spec-round.pdf`, 2019.