Romulus for Round 3

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

[1] Nagoya University (Japan) [2] NTU (Singapore) [3] NEC (Japan)

Abstract. Romulus is a family of lightweight authenticated encryption with associated data (AEAD) algorithms based on the extensively-studied ultra-lightweight tweakable block cipher (TBC) Skinny. It was designed to be small and efficient for constrained implementations while achieving 128-bit provable security based on standard security assumptions. In this note, we summarize new contents, analysis, benchmarks relative to Romulus.

In particular, we recall that Romulus is an all-round candidate, which presents excellent hardware performances and excellent software performances where lightweight cryptography makes the most sense (4-bit, 8-bit processors). In addition, Romulus has one primitive call of overhead for small messages (crucial for many lightweight applications), which is optimal. This performance profile does not prevent Romulus to be a very conservative candidate with absolutely no compromise on security regarding its mode (beyond-birthday-bound (BBB) security in contrary to most block cipher-based candidates, security proof in the standard model in contrary to most candidates, nonce-misuse resistance with graceful security degradation which is very important for constrained devices) or its internal primitive (very large security margin for Skinny, no attack nor distinguisher of any kind on its internal primitive unlike most sponge-based candidates, well understood and easy to analyse SPN construction). Moreover, Skinny is probably the most analysed internal primitive of the competition thus far (except current NIST standards AES or KECCAK). It has been incorporated in the French COVID-19 tracing application under French governmental security agency advice and is currently being considered for standardization at ISO/IEC.

Since the 2nd round selection, we provided several new contents for Romulus that we would like to incorporate in the submission if selected at 3rd round of the competition:

- we decided to simplify our submission by keeping only the versions based on Skinny-128/384 as we believe they are the most interesting (the nonce-respecting mode is now named Romulus-N, the nonce-misuse resistant mode is now named Romulus-M). Moreover, as publicly announced by the Skinny team, due to the huge security margin offered by Skinny-128/384, the number of rounds has been decreased from 56 to 40 (now named Skinny-128/384+), which still maintains more than 30% security margin even for near-brute force complexity distinguishers. This will directly offer a 40% boost on all hardware/software performance figures currently displayed for Romulus.
- we have added a simple and efficient rate-1 TBC-based hash function/XOF capability Romulus-H, by directly using Naito's provably-secure MDPH construction: the combination of the well-known Hirose DBL scheme and the Merkle-Damgård with Permutation (MDP) domain extender. We also designed leakage-resilient AEAD modes (Romulus-LR and Romulus-LR-TEDT for two levels of leakage resilience), both coming with a security proof in the leakage model. We emphasize that both these modes are using the same internal primitive Skinny-128/384+ and are themselves very similar (for example Romulus-LR simply re-injects the message in the tweak inputs compared to Romulus-N).
- we have added a proof for the INT-RUP security notion as well as the plaintext-awareness PA1 security notion for Romulus-M.
- we have provided threshold implementations of Romulus, which are very competitive according to publicly reported threshold implementations of other candidates. As shown by Naito *et al.* at Eurocrypt 2020, TBCs have a great small-size advantage over other designs for such implementations.
- we studied various performance trade-offs in hardware implementations, which show that the Romulus design offers excellent security-performance-area trade-offs.

In addition, we note that there is currently no TBC standard, while TBCs are very flexible primitives that can be used very easily to build various BBB secure modes.

Due to the 5-page limitation, the proofs, implementation details, etc. can be found in the companion papers submitted to the NIST LWC Workshop 2020 [12, 16].

1 Simpler and Faster Variants

The Skinny TBC [5] went through a lot of third-party analysis efforts over the past 4 years, with more than 30 cryptanalysis papers (and three cryptanalysis competitions conducted), which makes it the most analysed primitive of the competition, except AES or KECCAK. The number of attacked rounds stayed quite stable since the Skinny original publication. This confidence led to the incorporation of Skinny in the French COVID-19 tracing application under French governmental security agency advice. Moreover, Skinny is currently being considered for standardization at ISO/IEC (going on Committee Draft 18033-7).

At time of writing, the best known attacks against Skinny-128/384 cover 28 rounds [26] (out of 56 rounds), which means that the security margin is of 50% and actually much more if one considers only single-key attacks and/or attacks with a complexity lower than 2^{128}. Indeed, all these attacks have very high complexity, much more than 2^{200} in computational complexity and sometimes up to almost 2^{384}, and only work in the related-tweakey model where differences need to also be inserted in the tweak and/or key input. In the single-key model, the best known attacks against Skinny-128/384 covers only 22 rounds [9, 22, 24], again all these attacks having a very high computational complexity.

Compared to other block ciphers, where the security margin is usually at very best around 33%, this security margin is maybe too large. Even more so if we compare with permutations used in sponge functions proposals, where non-random behaviour can be usually exhibited for the full-round internal primitive for a complexity lower than the targeted security parameter of the whole scheme (actually often with practical complexity). For this reason, the Skinny team decided to propose a new variant of Skinny-128/384 (named Skinny-128/384+) by reducing its number of rounds from 56 to 40, to give a security margin of around 30% (in the worst-case related-tweakey scenario, without even excluding attacks with complexity much higher than 2^{128}), which still provides a very large security margin [23].

In order to simplify our NIST submission, we decided to only keep our first Romulus versions based on Skinny-128/384 (and thus now Skinny-128/384+), which were our original primary versions. Indeed, we believe these versions are the most interesting ones (for the same performance, they offer more flexibility) and in addition our submission gains in consistency as now **all** our modes are based on Skinny-128/384+ only. To summarize, Romulus will now consists in a nonce-respecting AEAD mode Romulus-N, a nonce-misuse resistant AEAD mode Romulus-M, two leakage resilient AEAD modes Romulus-LR, Romulus-LR-TEDT, and one hash function/XOF Romulus-H.

We emphasize that Romulus-N and Romulus-M are exactly the same as previous round Romulus-N1 and Romulus-M1, with simply Skinny-128/384+ used instead of Skinny-128/384. The security claims remain of course the same, while they will provide a 40% direct performance throughput/latency improvement over currently reported benchmarks (for the same area/memory). In Table 1, we summarize the specification of these new members and compare them with old variants.

Table 1: New and old variants of Romulus.

New members	Mode	Primitive	Comment
Romulus-N	Romulus-N1 [17, 18]		BBB nonce-respecting AEAD
Romulus-M	Romulus-M1 [17, 18]		BBB nonce-misuse resistant AEAD
Romulus-H	MDPH [20]	Skinny-128/384+	Hash function / XOF
Romulus-LR	AET-LR [12]		leakage resilient AEAD (CIML2 + CCAml1)
Romulus-LR-TEDT	TEDT [8]		leakage resilient AEAD (CIML2 + CCAmL2)

Previous members	Mode	Primitive	Comment
Romulus-N1		Skinny-128/384	BBB nonce-respecting AEAD
Romulus-N2	Romulus-N1 [17, 18]	Skinny-128/384	BBB nonce-respecting AEAD
Romulus-N3		Skinny-128/256	BBB nonce-respecting AEAD
Romulus-M1		Skinny-128/384	BBB nonce-misuse resistant AEAD
Romulus-M2	Romulus-M1 [17, 18]	Skinny-128/384	BBB nonce-misuse resistant
Romulus-M3		Skinny-128/256	BBB nonce-misuse resistant AEAD

2 Romulus-H: Hashing with Romulus

Since hashing capability was not originally added in the Romulus submission and since this can be achieved quite naturally, we have formalised Romulus-H: a hash function based on Skinny-128/384+. It is simply Skinny-128/384+ placed into Naito's MDPH construction [20], which consists of Hirose's Double-Block-Length (DBL) compression function [13] plugged into the Merkle-Damgård with Permutation (MDP) domain extender [14]. The full Romulus-H is depicted in Figure 1, while the formal specification can be found in the companion paper [16].

This construction is proven secure [20]: when the output is $2n$ bits, MDPH is indifferentiable [19] from a (variable-input-length) random oracle up to about $(n - \log n)$ queries [20] assuming the ideal (tweakable) block cipher. We set $n = 128$ and immediately have 121-bit indifferentiability. The standard reduction [4] tells that Romulus-H is proved to have 121-bit `atk`-security for any of $\texttt{atk} \in \{\texttt{collision}, \texttt{preimage}, \texttt{2nd-preimage}\}$. Note that indifferentiability is a very useful and versatile security notion for hash functions that some classical collision-resistant constructions fail

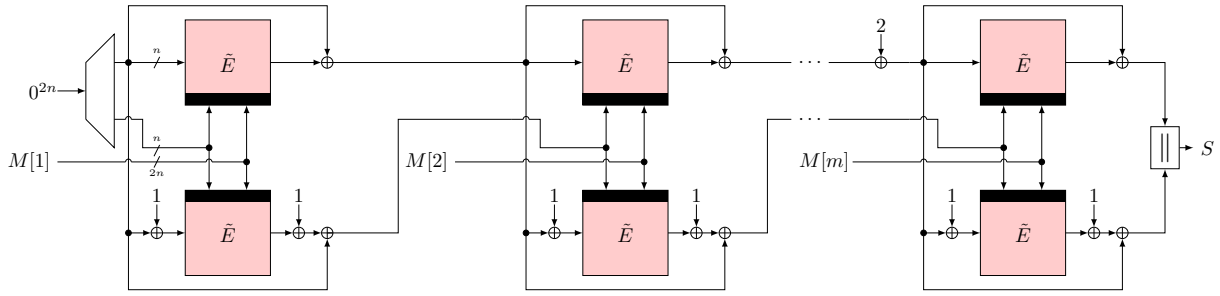

Fig. 1: Block diagram of Romulus-H hash function.

to meet [10]. Romulus-H can be easily turned into an eXtendable Output Function (XOF) that has an arbitrarily long output. This is because MDPH is indifferentiable from a (monolithic) random oracle, thus any black-box transformation that turns a RO into a XOF will also work for MDPH. One simple example is to use $H(M\|[0]), H(M\|[1]), \ldots,$ where H is the base hash function and $[i]$ denotes an encoding of integer i. In fact this is just a variant of standard MGF1 (Mask generation function). Additional computation cost of this transformation is small (one compression function call per block) thanks to the iterative nature of MDPH.

In terms of performances, Hirose's scheme requires two TBC calls, but since we are using Skinny-128/384+, 256-bit message blocks can be handled at each iteration, which makes Romulus-H an efficient rate-1 construction overall. We note that this is three times more efficient than the Skinny-Hash construction [6] (where Skinny-128/384 is used inside a standard sponge-based mode), for the same area cost. Moreover, the fact that each pair of TBC calls have the same tweakey input, combined with the lightweight tweakey schedule of Skinny, is helpful to achieve efficient implementations of this construction as the tweakey can be recovered and stored only once if only one Skinny-128/384+ hardware core is available. If two cores are available, they can share the same round keys. Moreover, we note that the hash can be naturally adapted to extremely constrained area environments by reducing the message input at every iteration (this is possible because Romulus-H places the message input in the tweak input of the TBC, and because Skinny-128/384+ tweakey schedule can be totally replaced by constants if some words are set to 0).

3 Romulus-LR and Romulus-LR-TEDT: Leakage-Resilient Modes for Romulus

Even though we provide efficient threshold implementations of Romulus, we studied how leakage resilience capability could be added to our candidate. It turns out that this can be achieved with a very simple modification of the Romulus mode. More precisely, we will propose two modes for leakage resilience: Romulus-LR and Romulus-LR-TEDT. More details on these modes and the corresponding security proofs can be found in a separate article [8, 12].

Romulus-LR is the first mode, which simply consists in (a) adding a key-derivation function (KDF) at the beginning of Romulus-N, to generate a temporary key K' that will be used in the subsequent TBC calls (b) re-injecting the message blocks inside the tweak input of each TBC call. It is then expected that the KDF and tag generating function (TGF), both using the master key K, should be properly protected with side-channels attacks countermeasures (such as masking). However, the long chain that depends on the message or associated data blocks can be left unprotected (or with much cheaper protection), which leads to a very efficient design (close to the original Romulus-N or Romulus-M). This mode, almost identical to Romulus-N, achieves the strong ciphertext integrity with misuse and leakage in the chosen-ciphertext model (CIML2) up to the birthday bound. It furthermore achieves integrity nonce-misuse resistance (MR-CINT) and integrity with the release of unverified plaintexts (INT-RUP) up to the birthday bound. Besides, it guarantees the nonce-misuse resilience of messages encrypted with fresh nonces, as long as the challenge queries are leak-free (CCAml1). In order to address even stronger adversaries, we offer Romulus-LR-TEDT.

Romulus-LR-TEDT is our second and most advanced leakage resilient mode, directly based on the provably secure TBC-based TEDT construction [8]. This Romulus-LR-TEDT mode basically consists in fine-tuning the details of TEDT to fit the advantages of Skinny-128/384+ and allow 128-bit nonce and long message/associated data inputs. TEDT provides full leakage resilience, that is, it limits the exploitability of physical leakages via side-channel attacks, even if these leakages happen during every message encryption and decryption operation. TEDT offers what is currently considered as the highest possible security notions in the presence of leakage, namely beyond birthday bound CIML2 and security against Chosen Ciphertext Attacks with nonce-misuse-resilience and Leakage (CCAmL2). While the initial TEDT proposal requires 4 TBC calls to process one n-bit message block, we optimize this to only 3 calls taking advantage of the properties of Skinny and our proposed hash function Romulus-H. This makes the performance more lightweight and closer to typical two-pass SIV-based schemes, which require 2 calls (except Romulus-M which requires

only 1.5 calls). Combined with its beyond birthday bound black-box and leakage-resilient security guarantees, it offers a great trade-off for sensitive applications.

Given Romulus-N, Romulus-M, Romulus-LR, Romulus-LR-TEDT, we believe our candidate offers variants that cover the whole spectrum of security levels and use-cases.

4 RUP Security of Romulus-M

Release of unverified plaintext (RUP) is a security notion introduced by Andreeva et al. [2]. It captures the scenario where the verification can leak the result of decryption (i.e., possibly an unauthentic plaintext) before the verification result is obtained. This is relevant in particular when the verifier's device has a limited amount of memory.

INT-RUP. Among several notions of RUP, authenticity/integrity under RUP, referred to as INT-RUP, is a popular notion for its importance and the possibility to achieve it without heavy constructions, such as the encode-then-encipher approach [7] with a wide-block primitive.

It is known that the generic SIV construction achieves INT-RUP security [3, Proposition 11]. However, in general, this does not extend to dedicated constructions based on SIV. We prove that Romulus-M is INT-RUP secure even though Romulus-M is not entirely based on SIV mainly due to the involvement of nonce in the encryption. The proof is fairly straightforward: due to the explicit domain separation we can let the TBC calls for encryption/decryption (and not authentication) freely accessible by the adversary and the problem reduces to the plain nonce-based MAC unforgeability. It shows strong authenticity of Romulus-M under RUP, both against nonce-respecting and nonce-misusing adversaries. The bounds are essentially the same as the regular authenticity bounds of Romulus-M, i.e., [18, Theorem 2] for nonce-respecting and [18, Theorem 3] for nonce-misusing adversaries.

Plaintext Awareness. The privacy notion in the RUP setting is called plaintext awareness [2]. Intuitively, it requires the existence of an extractor that can simulate the (unverified) decryption oracle without knowing the secret key. It has two versions, called PA1 and PA2, and the stronger notion of PA2 can be achieved only with a wide-block CCA-secure (tweakable) block cipher used in the encode-then-encipher approach (e.g., AEZ [15] or various TESs (Tweakable Enciphering Schemes)). It can be shown that Romulus-M is PA1 secure by following the proof of [3, Proposition 6], which proves that the scheme is PA1 secure if the MAC part is a PRF and the encryption part is PA1 secure. The MAC part of Romulus-M is a secure PRF, and the encryption part can be proved to be PA1 secure by following the proof of PA1 security of CBC mode and CTR mode [3, Proposition 12], with minor modifications to handle nonces.

5 New Hardware/Software and Threshold Implementations of Romulus, Comparisons

In order to show the design range of our candidate, we have recently provided new hardware implementations of Romulus, from round-based architectures to serial ones. Moreover, we also prepared threshold implementations. More details on these ASIC implementations and all measurements are given in the companion paper.

Round-Based Architecture. Round-based implementations are arguably the most interesting ones as they provide a high throughput with a reasonably low area, usually leading to the best trade-off for throughput/area ratio. The goal of the Romulus design is to have a very small area overhead on top of the underlying TBC, specially for these round-based implementations. In addition, Skinny was specifically designed to perform well with round-based implementations. In order to achieve that goal, we made sure that no costly Flip-Flops is required on top of the TBC and also that the number of possible inputs to each Flip-Flop and outputs of the circuits are minimized. Our results indicate that the best throughput/area trade-off is achieved by the `R2` architecture (which processes two rounds of the TBC per cycle), while the minimum energy is achieved by 4-round unrolling (the `R4` architecture). Romulus-N achieves 15 Gbps throughput with only about 8 KGE, which makes it an excellent candidate for high throughput and energy efficient architectures. This compares favourably to Ascon [11] and ACORN [25] (the winners of the CAESAR competition for "Lightweight applications" portfolio).

Serial Implementations. In case area minimisation is an absolute criterion, one can try to reduce the implementation data-path in a hope to obtain the lowest area possible. We have prepared byte-serial implementations of Romulus-N, that can reach area figures as low as 3.3 KGE (for a throughput close to 300 Mbps), which is an extremely low area for an AEAD mode with full n-bit security and standard model security proofs. Romulus-N is therefore an excellent candidate for low area and low power applications as well.

Threshold Implementations. We studied the 3-share threshold implementation of both the byte serial architecture and the single round architecture. The implementations are based on the threshold implementations provided by the Skinny team [5]. For only about 8 KGE and a throughput of ~8.5 Gbps, we can have a very efficient threshold implementation protected against side-channel attacks. This throughput/area ratio is competitive with even unprotected implementations of Ascon and ACORN. It can also be seen that the low area protected implementations of

Romulus-N1 and Romulus-N are very close in area to Ascon's unprotected low-area implementation and an order of magnitude faster. This comes from the fact (as explained in [21]) that while sponge-based constructions use a large permutation with a lot of non-linear operations, TBC-based schemes use a smaller permutation with cheaper and usually fully-linear key scheduling algorithms. This means that protecting the key scheduling algorithm is both cheaper and less demanding.

Software. Regarding software implementations, we applied the new fixslicing strategy [1] to Skinny which led to good performance results. Referring to the benchmarks from `https://lwc.las3.de/table.php`, we observe that for 32-bit platforms Romulus-N1 is generally placed in the middle of the rankings regarding throughput (Romulus-N being ranked in the first half). We remark that on these platforms, AES is already performing quite well. However, more interestingly, for very constrained platforms such as 8-bit architectures, Romulus-N1 ranks in the top tier, while Romulus-N would be among the top candidates (applying the 1.4 improvement ratio due to the reduction of the number of rounds). We believe these very constrained platforms (4-bit or 8-bit architectures) are probably the use-cases where lightweight cryptography makes the most sense in software.

References

1. Adomnicai, A., Najm, Z., Peyrin, T.: Fixslicing: A New GIFT Representation Fast Constant-Time Implementations of GIFT and GIFT-COFB on ARM Cortex-M. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020(3), 402–427 (2020)
2. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. In: Advances in Cryptology - ASIACRYPT 2014
3. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. IACR Cryptology ePrint Archive 2014, 144 (2014)
4. Andreeva, E., Mennink, B., Preneel, B.: Security reductions of the second round SHA-3 candidates. In: ISC 2010
5. 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 2016. Springer
6. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: SKINNY-AEAD and SKINNY-Hash. Submission to NIST LWC Project (2019)
7. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: ASIACRYPT. LNCS, vol. 1976, pp. 317–330. Springer (2000)
8. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.X.: Tedt, a leakage-resist aead mode for high physical security applications. IACR Transactions on Cryptographic Hardware and Embedded Systems 2020(1), 256–320 (Nov 2019)
9. Chen, Q., Shi, D., Sun, S., Hu, L.: Automatic Demirci-Selçuk Meet-in-the-Middle Attack on SKINNY with Key-Bridging. In: Zhou, J., Luo, X., Shen, Q., Xu, Z. (eds.) ICICS 2019. LNCS, vol. 11999, pp. 233–247. Springer (2019)
10. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: CRYPTO. pp. 430–448 (2005)
11. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Submission to Round 3 of the CAESAR competition (2016)
12. Guo, C., Khairallah, M., Peyrin, T.: AET-LR: Rate-1 Leakage-Resilient AEAD based on the Romulus Family. Submission to NIST LWC workshop 2020
13. Hirose, S.: Some plausible constructions of double-block-length hash functions. In: International Workshop on Fast Software Encryption. pp. 210–225. Springer (2006)
14. Hirose, S., Park, J.H., Yun, A.: A Simple Variant of the Merkle-Damgård Scheme with a Permutation. J. Cryptology 25(2), 271–309 (2012)
15. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015. LNCS, vol. 9056, pp. 15–44. Springer (2015)
16. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: New Results on Romulus. Submission to NIST LWC workshop 2020
17. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Romulus v1. Submission to NIST LWC Project (2019)
18. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Duel of the Titans: The Romulus and Remus Families of Lightweight AEAD Algorithms. IACR Transactions on Symmetric Cryptology (1) (2020), `https://eprint.iacr.org/2019/992`
19. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Theory of Cryptography, TCC 2004. pp. 21–39 (2004)
20. Naito, Y.: Optimally Indifferentiable Double-Block-Length Hashing Without Post-processing and with Support for Longer Key Than Single Block. In: LatinCrypt 2019. pp. 65–85. Springer (2019)
21. Naito, Y., Sasaki, Y., Sugawara, T.: Lightweight Authenticated Encryption Mode Suitable for Threshold Implementation. In: Advances in Cryptology - EUROCRYPT 2020. LNCS, vol. 12105. Springer (2020)
22. Shi, D., Sun, S., Derbez, P., Todo, Y., Sun, B., Hu, L.: Programming the Demirci-Selçuk Meet-in-the-Middle Attack with Constraints. In: Advances in Cryptology - ASIACRYPT 2018
23. teams, R..S.A.: New Romulus and SKINNY-AEAD variants. Announcement to the NIST lwc forum mailing list (13/05/20)
24. Tolba, M., Abdelkhalek, A., Youssef, A.M.: Impossible differential cryptanalysis of reduced-round SKINNY. In: AFRICACRYPT. LNCS, vol. 10239, pp. 117–134 (2017)
25. Wu, H.: ACORN: A Lightweight Authenticated Cipher v3. Submission to Round 3 of the CAESAR competition (2016)
26. Zhao, B., Dong, X., Meier, W., Jia, K., Wang, G.: Generalized Related-Key Rectangle Attacks on Block Ciphers with Linear Key Schedule: Applications to SKINNY and GIFT. Cryptology ePrint Archive, Report 2019/714 (2019)