A Practical Forgery Attack on Lilliput-AE

Orr Dunkelman[1,⋆], Nathan Keller[2,⋆⋆], Eran Lambooij[1,⋆], and Yu Sasaki[3]

[1] Computer Science Department, University of Haifa, Israel
orrd@cs.haifa.ac.il
[2] Department of Mathematics, Bar-Ilan University, Israel
nkeller@math.biu.ac.il
[3] NTT Secure Platform Laboratories, 3-9-11, Midori-cho Musashino-shi, Tokyo 180-8585, Japan
yu.sasaki.sk@hco.ntt.co.jp

Abstract. Lilliput-AE is a tweakable block cipher submitted as a candidate to the NIST lightweight cryptography standardization process. It is based upon the lightweight block cipher Lilliput, whose cryptanalysis so far suggests that it has a large security margin.
In this note we present an extremely efficient forgery attack on Lilliput-AE: Given a single arbitrary message of length about 2^{36} bytes, we can instantly produce another valid message that leads to the same tag, along with the corresponding ciphertext. The attack uses a weakness in the tweakey schedule of Lilliput-AE which leads to the existence of a related tweak differential characteristic with probability 1 in the underlying block cipher. The weakness we exploit, which does not exist in Lilliput, demonstrates the potential security risk in using a very simple tweakey schedule in which the same part of the key/tweak is re-used in every round, even when round constants are employed to prevent slide attacks. Following this attack, the Lilliput-AE submission to NIST was tweaked.

1 Introduction

Lilliput-AE is one of the 56 first round candidates for the NIST lightweight cryptography standardization process. It is an OCB based authenticated encryption scheme using the block cipher Lilliput [4] with a tweakey schedule. The predecessor Lilliput was cryptanalyzed in several papers (e.g., [9]); the best currently known attack on it faster than exhaustive key search targets only 17 of its 30 rounds, suggesting that it has a large security margin.

The Lilliput-AE family of ciphers contains both nonce-respecting and nonce-misuse resistant modes. It has 128-bit blocks and supports key sizes of 128, 192, and 256 bits. We focus on the nonce-misuse resistant mode and present two efficient forgery attacks. The first is a known message attack, in which given an arbitrary message of length at least $2^{32} + 2^{24} + 1$ blocks, we can instantly produce another message that leads to the same tag, along with its corresponding ciphertext. The second is a chosen message attack in which we ask for the encryption of a single message of length at least $2^{32} + 2^{24} + 2$ blocks in which we fix the values of two blocks, and then we instantly produce multiple messages that lead to the same tag. A variant of the known message attack in the

⋆ The first and third authors were supported in part by the Israel Ministry of Science and Technology, the Center for Cyber, Law, and Policy in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office and by the Israeli Science Foundation through grant No. 880/18.
⋆⋆ The second author was supported by the European Research Council under the ERC starting grant agreement n. 757731 (LightCrypt) and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

nonce respecting mode allows producing a pair of valid ciphertexts and tag that was not previously produced from the encryption oracle under a nonce that was used once (refer to [8] for the definition of nonce respecting forgery). All attacks succeed with probability 1.

The property of Lilliput-AE exploited in our attacks is a tiny detail in the tweakey schedule, that does not exist in the structure of Lilliput: one of the tweak words is reused without modification in all round subkeys, allowing us to introduce the same difference in all round subkeys by choosing an appropriate tweak difference. Due to the Feistel structure of the round function, this allows finding an iterative related-tweak differential with probability 1 for a single round; this differential persists for an arbitrary number of rounds, thus overcoming the conservative choice of a large number of rounds by the designers. The forgery is obtained by combining the differential with the way in which the internal block cipher is used in the authenticated encryption scheme.

The paradigm of using an extremely simple key schedule in which the same round key (or a few alternating round keys) is used in all encryption rounds, is used in numerous block ciphers (e.g., CRAFT [3], ITUbee [7], LED [6], Midori [2], Zorro [5]), in order to make key scheduling more lightweight. The potential security risk of this simplification with respect to related-key differential attacks was already discussed (e.g., in [5]). Some cipher designers (e.g., those of CRAFT, Midori, and Zorro) claim that the related-key scenario is practically irrelevant for the intended applications of their ciphers, and thus, this type of attacks is not a matter of concern.

The situation becomes more problematic in tweakable block ciphers, where related-tweak differences are a practical threat in most applications. Our attack demonstrates that in such scenarios, re-using the same part of the tweak in all rounds might lead to a practical break of the entire encryption scheme. Hence, simplifications of the tweakey schedule must be performed very cautiously.

2 Lilliput-AE

In this section we present the parts of the design of Lilliput-AE that are needed for understanding our attack. For the entire design, we refer the reader to [1].

Lilliput-AE uses a tweakable block cipher, Lilliput-TBC, with a 128-bit block, a 128-bit key and a 128-bit tweak (longer keys and tweaks are supported, and our attack applies to all of them). Lilliput-TBC is a generalized Feistel cipher with 32 to 42 rounds (depending on the exact version). As our attack is based on a probability 1 related-tweakey differential, we disregard the exact number of rounds. Each generalized Feistel round accepts two 64-bit halves, and treats them as 8 bytes each, as depicted in Figure 1.

The 64-bit round tweakey RTK^i is XORed to the inputs of the S-boxes (the F_i's). The values RTK^i are extracted from the tweakey schedule presented in Figure 2, which accepts two or three 64-bit words of tweak, and two, three or four 64-bit words of key. In each round, each of these words is updated separately, and all of them are XORed with the round number to form the round's subkey, RTK^i. The update of each of these words is done by applying a linear operation α_i to the word i (where α_0 is the identity function).

The authenticated encryption scheme has two modes: a nonce-respecting mode and a nonce-misuse resistant mode. The nonce-misuse resistant mode (on which we focus) is based on Deoxys' SCT. The associate data is encrypted (each block separately) and the outputs are XORed to obtain $Auth$. The Authentication of the message continues to XOR to $Auth$ the encryption of the different message blocks (after encryption), and the XOR of all blocks is then encrypted. Finally, the message encryption itself is done in a tweakable counter mode where the nonce is encrypted to generate a stream XORed to the message blocks. Figures 3a,4a and 4b present the exact mode (without

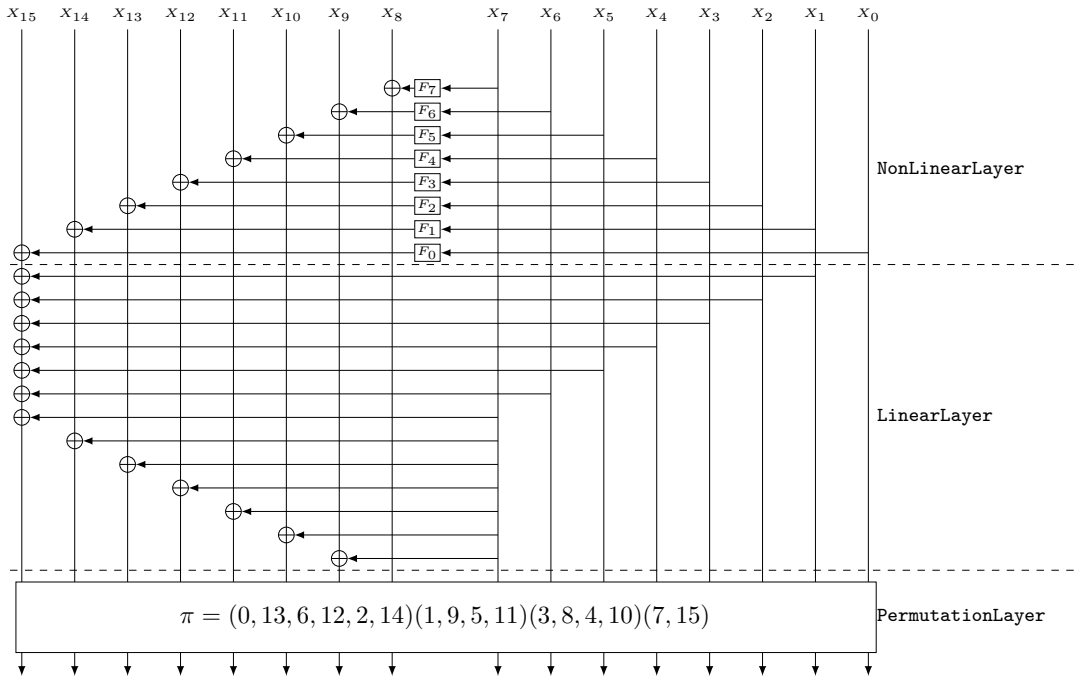


Fig. 1: Lilliput-TBC Round Function

padding). Obviously, different calls to the encryption function are domain-separated. As our attack is independent of the padding scheme, we do not discuss the padding rules used.

The nonce-respecting mode of Lilliput-AE is ΘCB3. The associate data is handled in the same way as in the nonce-misuse resistant mode, and each block is encrypted separately. The tag is computed by encrypting the XOR of all the message blocks, and then XORing the *Auth* of the associate data. See Figure 3a and 3b.

3 A Related Tweak Differential Attack on the Tweakable Block Cipher

The basic observation behind the attack is that in the tweakey schedule, the least significant word (i.e., 64 bits) of the tweak does not get updated in between rounds (as its update function α_0 is the identity function). As a result, if we consider two tweaks with difference of $\Delta = (\Delta_7, \ldots, \Delta_0)$ in the least significant word, then all corresponding round subkeys have difference Δ.

In order to use this property to obtain a 1-round iterative related-tweak differential characteristic with probability 1, we take the difference in the right hand side of the plaintext to be equal Δ, in order to nullify the input difference to all S-boxes, and thus, to pass the non-linear layer with probability 1. In order to cancel the effect of the linear layer, we make sure that $\Delta_7 = 0$, and that $\Delta_1 \oplus \ldots \oplus \Delta_6 = 0$. One possible way to achieve this is to take $\Delta_i = \Delta_j = \delta$ for some i, j, δ and

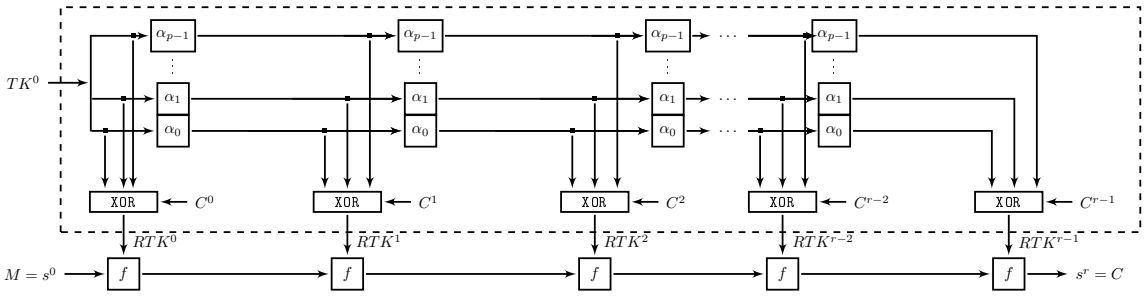
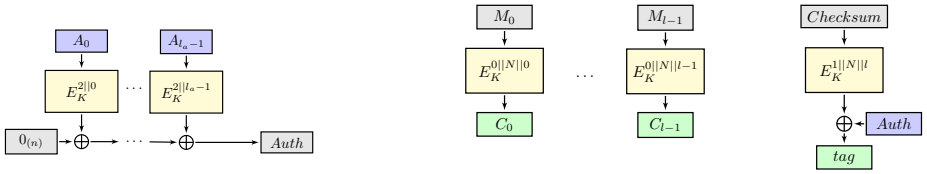


Fig. 2: Lilliput-TBC Tweakey Schedule



(a) Associated Data Processing

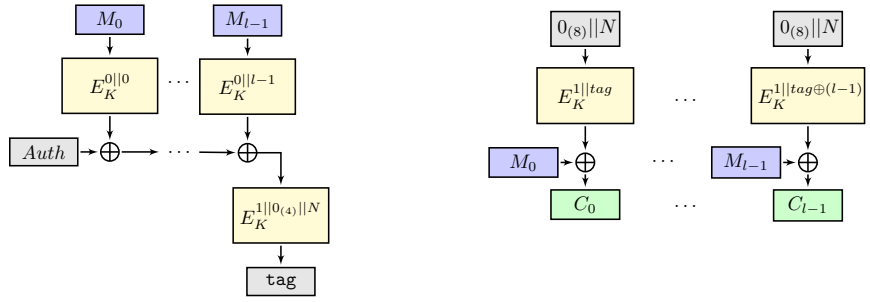
(b) Nonce-Respecting Encryption

Fig. 3: Lilliput-AE's Nonce-Respecting Mode of Operation (without Padding)

$\Delta_k = 0$ for all $k \neq i, j$. Finally, in order to cancel the effect of the permutation layer, we take the difference in the left half of the plaintext to be Δ', such that $\pi(\Delta, \Delta') = (\Delta, \Delta')$. An easy check shows that there are seven pairs (Δ, Δ') of this form, presented in Table 1. More precisely, for each of the cycles of π except for $(7, 15)$, we can take state difference of δ in the bytes of the cycle and 0 in the rest of the bytes, and tweak difference of δ in the bytes of the cycle that belong to the right half of the state, and 0 in the rest of the bytes (e.g., state difference of δ in bytes $3, 4, 8, 10$ and 0 in the rest of the bytes, and tweak difference of $(0, 0, 0, \delta, \delta, 0, 0, 0)$). We can also combine several cycles together, possibly with a different value of δ for each cycle.

State difference (bytes)	Tweak difference (bytes)
3, 4, 8, 10	3, 4
1, 5, 9, 11	1, 5
0, 2, 6, 12, 13, 14	0, 2, 6
1, 3, 4, 5, 8, 9, 10, 11	1, 3, 4, 5
0, 2, 3, 4, 6, 8, 10, 12, 13, 14	0, 2, 3, 4, 6
0, 1, 2, 5, 6, 9, 11, 12, 13, 14	0, 1, 2, 5, 6
0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14	0, 1, 2, 3, 4, 5, 6

Table 1: The seven configurations that lead to a probability 1 differential



(a) Authenticating Message Blocks (b) Encrypting Message Blocks

Fig. 4: Lilliput-AE's Nonce-Misuse Resistant Mode of Operation (without Padding)

Hence, we obtain multiple 1-round iterative related-tweak differential characteristics with probability 1. In the sequel, we use the characteristic with input and tweak differences

$$\Delta_P = (0,0,0,0,0,01_x,0,01_x||0,0,0,01_x,01_x,0,0,0), \qquad \Delta_T = (0,0,0,01_x,01_x,0,0,0).$$

4 Forgery Attacks on the Authenticated Encryption Scheme

The characteristic described above can be used to mount practical forgery attacks on Lilliput-AE. In this section we target the message part of the tag generation, but the attacks are similar for the Associated Data part of the tag generation.

Consider Lilliput-AE in the nonce-misuse resistant mode, whose structure is described above. Note that if we use the same nonce in two tag generations, then a collision before the last $E_K^{0||\ell-1}(M_{\ell-1})$ operation leads to a collision in `Tag`. We present two forgery attacks.

Chosen message forgery attack. By the structure of the message processing of the authenticated encryption scheme, if we take a message of length $l = 2^{32} + 2^{24} + 2$, then the blocks M_0, M_{l-2} are encrypted under the same key with tweaks whose XOR difference is $(0,0,0,01_x,01_x,0,0,0) = \Delta_T$, and the same goes for blocks M_1, M_{l-1}. Hence, if we take a message M of length l blocks, such that $M_{l-1} = M_1 \oplus \Delta_P$ and $M_{l-2} = M_0 \oplus \Delta_P$, where Δ_P is as defined above, then

$$E_K^{0||0}(M_0) \oplus E_K^{0||l-2}(M_{l-2}) = E_K^{0||1}(M_1) \oplus E_K^{0||l-1}(M_{l-1}) = \Delta_P. \tag{1}$$

Hence, the accumulated sum of the blocks $M_0, M_1, M_{l-2}, M_{l-1}$ is zero, which means that for every value of M_0, M_1 we obtain the same tag (assuming that the other message blocks remain unchanged). Hence, given a single chosen message, we can produce the tag of multiple (actually, as many as 2^{256}) messages we have not seen before.

We note that the difference Δ_P was chosen in order to minimize the length of the message l. Taking other cycles of the permutation π would have required a longer message until the corresponding tweak difference would appear in the counter. Longer messages can also be forged by the same attack – we just make the changes required for the forgery in the first $2^{32} + 2^{24} + 2$ blocks and leave the following blocks unchanged.

Known message forgery attack. We can obtain a known message forgery attack by intercepting the encryption of a single message M of length $l = 2^{32} + 2^{24} + 1$ (such that $l - 1 = \Delta_T$) and producing a forgery by a tailor-made modification of M. Denote the encryption and tag of M by $((C_0, C_1, \ldots, C_{l-1})||\texttt{tag})$. We take

$$M' = (M_{l-1} \oplus \Delta_P, M_1, \ldots, M_{l-2}, M_0 \oplus \Delta_P) \tag{2}$$

(i.e., we replace the first and last message blocks and leave the other blocks unchanged) and want to forge its encryption under the same key and nonce, and with the same associated data. We claim that the resulting ciphertext and tag is

$$((M_0 \oplus C_0 \oplus M_{l-1} \oplus \Delta, C_1, \ldots, C_{l-2}, M_{\Delta_T} \oplus C_{\Delta_T} \oplus M_0 \oplus \Delta)||\texttt{tag}).$$

By Figure 4a, it is clear that the ciphertext is correct, provided that the two messages indeed generate the same tag. To verify the tag collision, note that by the related tweak differential, we have

$$E_K^{0||0}(M_{l-1} \oplus \Delta_P) = E_K^{0||l-1}(M_{l-1}) \oplus \Delta_P,$$

and similarly,

$$E_K^{0||l-1}(M_0 \oplus \Delta_P) = E_K^{0||0}(M_0) \oplus \Delta_P,$$

and thus, the contributions of the first and last message blocks to the accumulated sum cancel, and hence, there is a tag collision between the two encryptions.

Nonce respecting mode. The known message forgery attack described above can be applied also in the nonce respecting mode. That is, given the encryption of a message M of length $2^{32} + 2^{24} + 1$, one can compute the encryption and tag of the message M' (as defined in (2)) under the same key and nonce, with the same associated data.

Possible tweak. Since our attack crucially depends on the ability to produce the same difference in all round keys, it would be thwarted by replacing the identity transformation α_0 used in the tweakey schedule with some mixing linear transformation unequal to the other α_i's. Following our report, the authors of Lilliput-AE indeed accepted this proposal.

Acknowledgements

We are grateful to the Lilliput-AE team for confirming our findings and for allowing us to use the figures from the specification document in this note.

References

1. A. Adomnicai, T. P. Berger, C. Clavier, J. Francq, P. Huynh, V. Lallemand, K. Le Gouguec, M. Minier, L. Reynaud, and G. Thomas, *Lilliput-AE: a new lightweight tweakable block cipher for authenticated encryption with associated data*, Submission to the NIST Lightweight Cryptography Standardization Process, 2019. Available at: `https://csrc.nist.gov/Projects/Lightweight-Cryptography`.
2. S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni, *Midori: A Block Cipher for Low Energy*, proceedings of ASIACRYPT 2015, Lecture Notes in Computer Science 9453, pp. 411–436, Springer, 2015.

3. C. Beierle, G. Leander, A. Moradi, and S. Rasoolzadeh, *CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks*, *IACR Transactions on Symmetric Cryptology* vol. 2019, no. 1, pp. 5–45, 2019.
4. T. P. Berger, J. Francq, M. Minier, and G. Thomas, *Extended generalized Feistel networks using matrix representation to propose a new lightweight block cipher: Lilliput*, *IEEE Trans. Computers* **65(7)** (2016), pp. 2074–2089.
5. B. Gérard, V. Grosso, M. Naya-Plasencia, and F.-X. Standaert, *Block ciphers that are easier to mask: How far can we go?*, proceedings of CHES 2013, Lecture Notes in Computer Science 8086, pp. 383–399, Springer, 2013.
6. J. Guo, T. Peyrin, A. Poschmann, and M.J.B. Robshaw, *The LED Block Cipher*, proceedings of CHES 2011, Lecture Notes in Computer Science 6917, pp. 326–341, Springer, 2011.
7. F. Karakoç, H. Demirci., and A.E. Harmancı, *ITUbee: A Software Oriented Lightweight Block Cipher*, proceedings of Lightsec 2013, Lecture Notes in Computer Science 8162, pp. 16–27, Springer, 2013.
8. P. Rogaway, *Nonce-based symmetric encryption*, proceedings of FSE 2004, Lecture Notes in Computer Science 3017, pp. 348–359, Springer, 2004.
9. Y. Sasaki and Y. Todo, *Tight bounds of differentially and linearly active S-Boxes and division property of Lilliput*, *IEEE Trans. Computers* **67(5)** (2018), pp. 717–732.