From: orr.dunkelman@gmail.com
Sent: Monday, July 8, 2019 7:00 AM
To: lightweight-crypto
Subject: Official comment: Lilliput-AE
Attachments: report.pdf

Dear all,

During the last few weeks we have worked on the candidate Lilliput-AE. During this work we have identified a probability 1 related-tweakey differential characteristic that allows mounting a practical forgery attack in the nonce-misuse resistant model (as well as producing a valid ciphertext+tag in the nonce respecting model).

We have disclosed the observations to the designers who confirmed our findings, and they are working on a fix to the problem (that can be easily solved by changing the \alpha_0 component of the design).

It is worth noting that the original Lilliput cipher (proposed at IEEE Trans. Computers Vol.65 No.7 2016) is not affected by the attack.

The full technical details can be found in the attached PDF.

Nathan, Yu, Eran, and Orr.

--
Orr Dunkelman,
Orr.Dunkelman@gmail.com

A Related Tweak Differential Attack on Full Lilliput-AE

Orr Dunkelman, Nathan Keller, Eran Lambooij, and Yu Sasaki

July 4, 2019

1 Introduction

Lilliput-AE is one of the 56 first round candidates for the NIST Lightweight cryptography competition. It is a OCB based authenticated encryption scheme using the block cipher Lilliput with a tweakey schedule. The Lilliput-AE family of ciphers contains both nonce-respecting and nonce-misuse resistant modes and can be instantiated with key sizes of 128, 192, and 256 bits. The block size is 128 bits.

We propose a related tweak differential attack on the full block cipher. This attack translates to a forgery attack on the authenticated encryption scheme, in the nonce-misuse resistant mode. The attack needs a message of 2^{32} blocks to obtain a forgery with probablility 1. In the nonce respecting mode, the attack allows producing valid ciphertexts and tags of messages that weren't queried, under a nonce that was used once (which is considered a forgery, according to [Phillip Rogaway, Nonce-Based Symmetric Encryption, FSE 2004, pp. 348-359]).

2 Lilliput-AE

See `https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/LILLIPUT-AE-spec.pdf` for the complete description of Lilliput-AE.

3 Related Tweak Differential attack

The related-tweak differential characteristic used in this paper is a 1-round iterative differential characteristic with probability 1. We make use of the fact that 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) . This leads to the observation that if we introduce a difference in the tweak, this difference is XORed to the state before the non-linear layer in each round. We use this observation to cancel the differences that go into the non-linear layer, to obtain a probability 1 differential characteristic.

The characteristic is built as follows: We introduce difference Δ into bytes X_3, X_4, X_8 and X_{10} of the message and bytes X_3 and X_4 of the tweak. Due to the XOR of the tweak (which has difference $(0, 0, 0, \Delta, \Delta, 0, 0, 0)$), the difference going into the non-linear part of the round function

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 on the Lilliput block cipher.

is 0. Since two bytes are active and X_7 is inactive, the difference introduced by the linear part of the round function is also 0. Now the permutation used in the Lilliput round function maps: $X_3 \to X_8$, $X_4 \to X_{10}$, $X_8 \to X_4$, and $X_{10} \to X_3$, and thus, after the round function we have a Δ difference in bytes X_3, X_4, X_8 and X_{10} of the state. There exist seven such configurations that lead to an iterative related tweak differential characteristic with probability 1 (see Table 1).

4 Forgery attacks

The characteristic described in section 3 can be used to mount forgery attacks on Lilliput-AE in the nonce-misuse resistant mode. In this section we target the message part of the tag generation, but the attacks are similar for the Auxiliary Data part of the tag generation.

Consider Lilliput-AE in the nonce-misuse resistant mode. Let $E_K^T(X)$ denote the encryption with key K and tweak T, then given some state `Auth` and message blocks $M_0, M_1, M_2, \ldots, M_{\ell-1}$ we compute the tag as

$$\texttt{Tag}' = \texttt{Auth} \oplus E_K^{0||0}(M_0) \oplus E_K^{0||1}(M_1) \oplus E_K^{0||2}(M_2) \oplus \ldots \oplus E_K^{0||\ell-1}(M_{\ell-1})$$

and for nonce N,

$$\texttt{Tag} = E_K^{1||0_{(4)}||N}(\texttt{Tag}')$$

Note that if we use the same nonce in two tag generations, then a collision in $\texttt{Tag}'$ leads to a collision in `Tag`.

We present two different forgery attacks.

First forgery attack. Let Δ be the message difference inducing the probability 1 characteristic and let $\Delta_T = 0||\Delta T'$ the tweak difference. If we now take the any M_0, M_1 and $M_{\Delta_T} = M_0 \oplus \Delta$ and $M_{\Delta_T+1} = M_1 \oplus \Delta$, then

$$E_K^{0||0}(M_0) \oplus E_K^{\Delta_T}(\Delta \oplus M_0) = E_K^{0||1}(M_1) \oplus E_K^{\Delta_T \oplus 1}(\Delta \oplus M_1) = \Delta. \quad (1)$$

By Equation 1, the influence of these four message blocks is 0, which means that for every M_0, M_1 we get the same tag. This allows us to construct a tag for messages we have not seen before.

The cost of this attack is 1 query of length $\Delta_T + 1$ blocks, which (for the tweak difference `0x01` in bytes 3 and 4) is approximately 2^{32}. This attack works on all key sizes.

Second forgery attack. We can ask for the encryption of any single message of size $\Delta_T + 1$ and generate another message that leads to the same tag, along with its ciphertext. Indeed, reusing the notations of the first forgery attack, we denote the ciphertext/tag of a message $M = (M_0, M_1, \ldots, M_{\Delta_T})$, encrypted under the key K and the nonce N, with associated data A, by $(C_0, C_1, \ldots, C_{\Delta_T}), T$. Consider the encryption of the message

$$M' = (M_{\Delta_T} \oplus \Delta, M_1, \ldots, M_{\Delta_T - 1}, M_0 \oplus \Delta)$$

under the same key and nonce, and with the same associated data. We claim that the resulting ciphertext/tag is

$$(M_0 \oplus C_0 \oplus M_{\Delta_T} \oplus \Delta, C_1, \ldots, C_{\Delta_T - 1}, M_{\Delta_T} \oplus C_{\Delta_T} \oplus M_0 \oplus \Delta), T.$$

It is clear that the ciphertext is correct, provided that the two messages indeed generate the same tag. To verify this, note that by the probability 1 differential, we have

$$E_K^{0||0}(M_{\Delta_T} \oplus \Delta) = E_K^{\Delta_T}(M_{\Delta_T}) \oplus \Delta,$$

and similarly,

$$E_K^{\Delta_T}(M_0 \oplus \Delta) = E_K^{0||0}(M_0) \oplus \Delta,$$

and thus, the two encryption processes collide on `Tag'`, and hence, collide on `Tag` as well.

Nonce respecting mode. The attack described above does not apply (as is) in the nonce respecting mode, since in that mode, in the message processing phase, the difference between two nonces must affect words of the tweak that are updated between rounds, and so one cannot construct a differential with probability 1 in that part. We note however that after asking for the encryption of a single message of length slightly more than 2^{32}, an adversary is able to produce valid tags and ciphertexts for many other messages under the same nonce.

5 Possible tweak

There are various possible tweaks that will be sufficient for thwarting the attack presented above. It seems that the simplest one is replacing the identity transformation α_0 with some mixing linear transformation (like α_1, α_2, etc., but not identical to one of them).

From: orr.dunkelman@gmail.com
Sent: Thursday, July 11, 2019 2:56 PM
To: lwc-forum@list.nist.gov
Subject: [lwc-forum] Official comment: Lilliput-AE
Attachments: report.pdf

Dear all,

During the last few weeks we have worked on the candidate Lilliput-AE. During this work we have identified a probability 1 related-tweakey differential characteristic that allows mounting a practical forgery attack in the nonce-misuse resistant model (as well as producing a valid ciphertext+tag in the nonce respecting model).

We have disclosed the observations to the designers who confirmed our findings, and they are working on a fix to the problem (that can be easily solved by changing the \alpha_0 component of the design).

It is worth noting that the original Lilliput cipher (proposed at IEEE Trans. Computers Vol.65 No.7 2016) is not affected by the attack.

The full technical details can be found in the attached PDF.

Nathan, Yu, Eran, and Orr.

--
Orr Dunkelman,
Orr.Dunkelman@gmail.com

A Related Tweak Differential Attack on Full Lilliput-AE

Orr Dunkelman, Nathan Keller, Eran Lambooij, and Yu Sasaki

July 4, 2019

1 Introduction

Lilliput-AE is one of the 56 first round candidates for the NIST Lightweight cryptography competition. It is a OCB based authenticated encryption scheme using the block cipher Lilliput with a tweakey schedule. The Lilliput-AE family of ciphers contains both nonce-respecting and nonce-misuse resistant modes and can be instantiated with key sizes of 128, 192, and 256 bits. The block size is 128 bits.

We propose a related tweak differential attack on the full block cipher. This attack translates to a forgery attack on the authenticated encryption scheme, in the nonce-misuse resistant mode. The attack needs a message of 2^{32} blocks to obtain a forgery with probablility 1. In the nonce respecting mode, the attack allows producing valid ciphertexts and tags of messages that weren't queried, under a nonce that was used once (which is considered a forgery, according to [Phillip Rogaway, Nonce-Based Symmetric Encryption, FSE 2004, pp. 348-359]).

2 Lilliput-AE

See `https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/LILLIPUT-AE-spec.pdf` for the complete description of Lilliput-AE.

3 Related Tweak Differential attack

The related-tweak differential characteristic used in this paper is a 1-round iterative differential characteristic with probability 1. We make use of the fact that 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) . This leads to the observation that if we introduce a difference in the tweak, this difference is XORed to the state before the non-linear layer in each round. We use this observation to cancel the differences that go into the non-linear layer, to obtain a probability 1 differential characteristic.

The characteristic is built as follows: We introduce difference Δ into bytes X_3, X_4, X_8 and X_{10} of the message and bytes X_3 and X_4 of the tweak. Due to the XOR of the tweak (which has difference $(0, 0, 0, \Delta, \Delta, 0, 0, 0)$), the difference going into the non-linear part of the round function

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 on the Lilliput block cipher.

is 0. Since two bytes are active and X_7 is inactive, the difference introduced by the linear part of the round function is also 0. Now the permutation used in the Lilliput round function maps: $X_3 \to X_8$, $X_4 \to X_{10}$, $X_8 \to X_4$, and $X_{10} \to X_3$, and thus, after the round function we have a Δ difference in bytes X_3, X_4, X_8 and X_{10} of the state. There exist seven such configurations that lead to an iterative related tweak differential characteristic with probability 1 (see Table 1).

4 Forgery attacks

The characteristic described in section 3 can be used to mount forgery attacks on Lilliput-AE in the nonce-misuse resistant mode. In this section we target the message part of the tag generation, but the attacks are similar for the Auxiliary Data part of the tag generation.

Consider Lilliput-AE in the nonce-misuse resistant mode. Let $E_K^T(X)$ denote the encryption with key K and tweak T, then given some state `Auth` and message blocks $M_0, M_1, M_2, \ldots, M_{\ell-1}$ we compute the tag as

$$\texttt{Tag}' = \texttt{Auth} \oplus E_K^{0||0}(M_0) \oplus E_K^{0||1}(M_1) \oplus E_K^{0||2}(M_2) \oplus \ldots \oplus E_K^{0||\ell-1}(M_{\ell-1})$$

and for nonce N,

$$\texttt{Tag} = E_K^{1||0_{(4)}||N}(\texttt{Tag}')$$

Note that if we use the same nonce in two tag generations, then a collision in $\texttt{Tag}'$ leads to a collision in `Tag`.

We present two different forgery attacks.

First forgery attack. Let Δ be the message difference inducing the probability 1 characteristic and let $\Delta_T = 0||\Delta T'$ the tweak difference. If we now take the any M_0, M_1 and $M_{\Delta_T} = M_0 \oplus \Delta$ and $M_{\Delta_T+1} = M_1 \oplus \Delta$, then

$$E_K^{0||0}(M_0) \oplus E_K^{\Delta_T}(\Delta \oplus M_0) = E_K^{0||1}(M_1) \oplus E_K^{\Delta_T \oplus 1}(\Delta \oplus M_1) = \Delta. \quad (1)$$

By Equation 1, the influence of these four message blocks is 0, which means that for every M_0, M_1 we get the same tag. This allows us to construct a tag for messages we have not seen before.

The cost of this attack is 1 query of length $\Delta_T + 1$ blocks, which (for the tweak difference `0x01` in bytes 3 and 4) is approximately 2^{32}. This attack works on all key sizes.

Second forgery attack. We can ask for the encryption of any single message of size $\Delta_T + 1$ and generate another message that leads to the same tag, along with its ciphertext. Indeed, reusing the notations of the first forgery attack, we denote the ciphertext/tag of a message $M = (M_0, M_1, \ldots, M_{\Delta_T})$, encrypted under the key K and the nonce N, with associated data A, by $(C_0, C_1, \ldots, C_{\Delta_T}), T$. Consider the encryption of the message

$$M' = (M_{\Delta_T} \oplus \Delta, M_1, \ldots, M_{\Delta_T - 1}, M_0 \oplus \Delta)$$

under the same key and nonce, and with the same associated data. We claim that the resulting ciphertext/tag is

$$(M_0 \oplus C_0 \oplus M_{\Delta_T} \oplus \Delta, C_1, \ldots, C_{\Delta_T - 1}, M_{\Delta_T} \oplus C_{\Delta_T} \oplus M_0 \oplus \Delta), T.$$

It is clear that the ciphertext is correct, provided that the two messages indeed generate the same tag. To verify this, note that by the probability 1 differential, we have

$$E_K^{0||0}(M_{\Delta_T} \oplus \Delta) = E_K^{\Delta_T}(M_{\Delta_T}) \oplus \Delta,$$

and similarly,

$$E_K^{\Delta_T}(M_0 \oplus \Delta) = E_K^{0||0}(M_0) \oplus \Delta,$$

and thus, the two encryption processes collide on `Tag'`, and hence, collide on `Tag` as well.

Nonce respecting mode. The attack described above does not apply (as is) in the nonce respecting mode, since in that mode, in the message processing phase, the difference between two nonces must affect words of the tweak that are updated between rounds, and so one cannot construct a differential with probability 1 in that part. We note however that after asking for the encryption of a single message of length slightly more than 2^{32}, an adversary is able to produce valid tags and ciphertexts for many other messages under the same nonce.

5 Possible tweak

There are various possible tweaks that will be sufficient for thwarting the attack presented above. It seems that the simplest one is replacing the identity transformation α_0 with some mixing linear transformation (like α_1, α_2, etc., but not identical to one of them).

From: julien francq <julien_francq@yahoo.fr>
Sent: Friday, July 12, 2019 12:38 PM
To: lwc-forum@list.nist.gov; orr.dunkelman@gmail.com; lightweight-crypto
Subject: [lwc-forum] Official comment: Lilliput-AE
Attachments: report.pdf

Dear all,

First, we would like to thank Nathan, Yu, Eran, and Orr for their security analysis of Lilliput-AE.

Our first analysis tends to confirm the described security flaw.

We are going to evaluate the minor algorithm changes the authors propose, and once we checked completely the impacts from a security and performance point of view, we'll keep the community posted.

Our goal is to submit a completely renewed submission package (specifications, source code, hardware implementations) in one week.

Best regards,

The Lilliput-AE team.

Le jeudi 11 juillet 2019 à 20:56:02 UTC+2, <orr.dunkelman@gmail.com> a écrit :

Dear all,

During the last few weeks we have worked on the candidate Lilliput-AE. During this work we have identified a probability 1 related-tweakey differential characteristic that allows mounting a practical forgery attack in the nonce-misuse resistant model (as well as producing a valid ciphertext+tag in the nonce respecting model).

We have disclosed the observations to the designers who confirmed our findings, and they are working on a fix to the problem (that can be easily solved by changing the \alpha_0 component of the design).

It is worth noting that the original Lilliput cipher (proposed at IEEE Trans. Computers Vol.65 No.7 2016) is not affected by the attack.

The full technical details can be found in the attached PDF.

Nathan, Yu, Eran, and Orr.

--
Orr Dunkelman,
Orr.Dunkelman@gmail.com

From: julien francq <julien_francq@yahoo.fr>
Sent: Saturday, July 20, 2019 12:09 PM
To: lwc-forum@list.nist.gov; orr.dunkelman@gmail.com; lightweight-crypto
Subject: Official Comment: Lilliput-AE
Attachments: LilliputAE.zip

Dear all,

First of all, we would like to thank once again Nathan, Yu, Eran and Orr for their security analysis of Lilliput-AE.
We can now definitely confirm that their findings are correct.

We have also analyzed the impacts of their proposed security patch, and we can now also confirm that it efficiently corrects the flaw, at the expense of very limited performance overhead. For example, Lilliput-AE is still very competitive in hardware and software compared to Ascon and Acorn.

You can find in attached file the new Lilliput-AE submission package folder (v1.1). The Appendix section of the Lilliput-AE specifications describes the complete changelog. The folder will also be available soon on the webpage:
https://paclido.fr/lilliput-ae/

Moreover, we have taken the opportunity of this new package edition to submit a new implementation version of the Lilliput-AE padding scheme. It was correct in the documentation but not in the source codes: this discrepancy has been corrected for every implementation (C, Python, VHDL). Further details are provided in a dedicated changelog file under REFERENCE_IMPLEMENTATION/CHANGELOG.txt.

We thank in advance further third-party analysis of this new Lilliput-AE version.

Best regards,

The Lilliput-AE team.