From: hash-forum@nist.gov on behalf of Hirotaka Yoshida [hirotaka.yoshida.qv@hitachi.com]
Sent: Friday, May 29, 2009 7:38 AM
To: Multiple recipients of list
Subject: OFFICIAL COMMENT: Lesamnta

Dear NIST, all,

In round 1 technical evaluation, NIST intends to perform an efficiency analysis on Intel Core 2 Duo Processor.

Given the importance of Intel Processors, we think that it is reasonable to consider hash algorithm performance using a new set of AES instructions to be introduced into the next generation of Intel Processors.

We would like to explain why AES instructions should be considered:
Based on our observation that Intel CPU performance figures of several SHA-3 candidates are due to the use of instructions in SSE, we think that new version of SSE which employs AES instructions should be considered as the same way as the current version of it. We expect that Intel CPUs with AES instructions will be widely used by the end of the SHA-3 competition.

If AES instructions are considered, we would suggest to use the following known methods to evaluate performance with them:
a) Implement using certain compliers GAS assembler or Intel compiler
   This depends on the language, assembly or C.
   This method is described in an Intel website [1].
b) Confirm the correctness of test vectors using the Intel SDE simulator
   which makes use of AES instructions
   This method is also described in [1].
c) Speed measurement of the implementation where the AESENC instruction
   is replaced with the PMULUDQ instruction.
   AESENC and PMULUDQ have the same throughput and same latency.
   This method is proposed by the Vortex team.

We evaluate the performance of Lesamnta using the above method.
The results are listed:
http://www.sdl.hitachi.co.jp/crypto/lesamnta/

Reference
[1]http://software.intel.com/en-us/blogs/2008/08/11/emulation-of-new-instructions/

Best regards,
Hirotaka Yoshida

From: hash-forum@nist.gov on behalf of Hirotaka Yoshida [hirotaka.yoshida.qv@hitachi.com]
Sent: Friday, May 29, 2009 9:33 AM
To: Multiple recipients of list
Subject: OFFICIAL COMMENT: Lesamnta

Dear NIST, dear all,

This is a response to NIST's statement that in round 1 technical evaluation, NIST invites the public to compare results on additional platforms (e.g., 8-bit processors, etc.)

We compare the implementation costs of various SHA-3 candidates on low-cost 8-bit CPUs by estimating RAM/ROM requirements of them.
The PDF of our work can be found on the following URL:
http://www.sdl.hitachi.co.jp/crypto/lesamnta/A_Study_on_RAM_Requirements.pdf
We would be grateful if this work is considered.

Hereafter, we would like to describe reasons why considering 8-bit CPUs is important for hash algorithm implementations.
Firstly, 8-bit CPUs are really popular, which is based on a report [1] saying that about 55 % of all CPUs sold in the world are 8-bitmicrocontrollers and microprocessors.
Scondly, there is a important tradeoff which should be made between cryptographic functionality and the cost of the device. Even some symmetric cryptographic algorithms could be too expensive for some applications using low-end smart cards and RFID tags which typically employ low-cost 8-bit CPU.
Thirdly, it is thinkable that in the near future, we will see a wide variety of security applications using low-cost 8-bit CPUs such as wireless sensor network, which is based on the report in [2] saying that passive RFID tag market is expected to hit $486M in 2013,

Considering all this above, we expect that RAM/ROM requirements on low-cost 8-bit CPUs should be considered as an important factor in comparison of SHA-3 candidates.

References
[1]http://en.wikipedia.org/wiki/Microprocessor
[2]http://www.infoworld.com/t/networking/passive-rfid-tag-market-hit-486m-in-2013-102

Please note that the authors are involved with the submission of a SHA-3 candidate Lesamnta. This is a view from these people.

Best regards,
Hirotaka Yoshida

Caswell, Sara J.

From: hash-forum@nist.gov on behalf of Colin B [mesadesign@colinb.cts.com]
Sent: Monday, June 08, 2009 12:36 AM
To: Multiple recipients of list
Subject: Re: OFFICIAL COMMENT: Lesamnta

Your reasoning is good, but why study only a subset of the submissions and why that particular subset?

---- <hash-forum@nist.gov> wrote:
>
>
> Dear NIST, dear all,
>
> This is a response to NIST's statement that in round 1 technical
> evaluation, NIST invites the public to compare results on additional
> platforms (e.g., 8-bit processors, etc.)
>
> We compare the implementation costs of various SHA-3 candidates on
> low-cost 8-bit CPUs by estimating RAM/ROM requirements of them.
> The PDF of our work can be found on the following URL:
> http://www.sdl.hitachi.co.jp/crypto/lesamnta/A_Study_on_RAM_Requiremen
> ts.pdf We would be grateful if this work is considered.
>
> Hereafter, we would like to describe reasons why considering 8-bit
> CPUs is important for hash algorithm implementations.
> Firstly, 8-bit CPUs are really popular, which is based on a report [1]
> saying that about 55 % of all CPUs sold in the world are
> 8-bitmicrocontrollers and microprocessors.
> Scondly, there is a important tradeoff which should be made between
> cryptographic functionality and the cost of the device. Even some
> symmetric cryptographic algorithms could be too expensive for some
> applications using low-end smart cards and RFID tags which typically
> employ low-cost 8-bit CPU.
> Thirdly, it is thinkable that in the near future, we will see a wide
> variety of security applications using low-cost 8-bit CPUs such as
> wireless sensor network, which is based on the report in [2] saying
> that passive RFID tag market is expected to hit $486M in 2013,
>
> Considering all this above, we expect that RAM/ROM requirements on
> low-cost 8-bit CPUs should be considered as an important factor in
> comparison of SHA-3 candidates.
>
> References
> [1]http://en.wikipedia.org/wiki/Microprocessor
> [2]http://www.infoworld.com/t/networking/passive-rfid-tag-market-hit-4
> 86m-in-2013-102
>
> Please note that the authors are involved with the submission of a
> SHA-3 candidate Lesamnta. This is a view from these people.
>
> Best regards,
> Hirotaka Yoshida
>
>
>

Caswell, Sara J.

From: hash-forum@nist.gov on behalf of Hirotaka Yoshida [hirotaka.yoshida.qv@hitachi.com]
Sent: Monday, June 08, 2009 6:46 AM
To: Multiple recipients of list
Subject: Re: OFFICIAL COMMENT: Lesamnta

Thank you for your interest.

This is simply because that we had limited resources and we did not
have time enough to study all the submissions by 1 st of June (The NIST deadline for comment and analysis). We hope that other
research group would extend our study to a study on all submissions.

Best regards,
Hirotaka Yoshida

> Your reasoning is good, but why study only a subset of the submissions and why that particular subset?
>
> ---- <hash-forum@nist.gov> wrote:
>>
>> Dear NIST, dear all,
>>
>> This is a response to NIST's statement that in round 1 technical
>> evaluation, NIST invites the public to compare results on additional
>> platforms (e.g., 8-bit processors, etc.)
>>
>> We compare the implementation costs of various SHA-3 candidates on
>> low-cost 8-bit CPUs by estimating RAM/ROM requirements of them.
>> The PDF of our work can be found on the following URL:
>> http://www.sdl.hitachi.co.jp/crypto/lesamnta/A_Study_on_RAM_Requireme
>> nts.pdf We would be grateful if this work is considered.
>>
>> Hereafter, we would like to describe reasons why considering 8-bit
>> CPUs is important for hash algorithm implementations.
>> Firstly, 8-bit CPUs are really popular, which is based on a report
>> [1] saying that about 55 % of all CPUs sold in the world are
>> 8-bitmicrocontrollers and microprocessors.
>> Scondly, there is a important tradeoff which should be made between
>> cryptographic functionality and the cost of the device. Even some
>> symmetric cryptographic algorithms could be too expensive for some
>> applications using low-end smart cards and RFID tags which typically
>> employ low-cost 8-bit CPU.
>> Thirdly, it is thinkable that in the near future, we will see a wide
>> variety of security applications using low-cost 8-bit CPUs such as
>> wireless sensor network, which is based on the report in [2] saying
>> that passive RFID tag market is expected to hit $486M in 2013,
>>
>> Considering all this above, we expect that RAM/ROM requirements on
>> low-cost 8-bit CPUs should be considered as an important factor in
>> comparison of SHA-3 candidates.
>>
>> References
>> [1]http://en.wikipedia.org/wiki/Microprocessor
>> [2]http://www.infoworld.com/t/networking/passive-rfid-tag-market-hit-
>> 486m-in-2013-102
>>
>> Please note that the authors are involved with the submission of a
>> SHA-3 candidate Lesamnta. This is a view from these people.
>>

From: hash-forum@nist.gov on behalf of Hirotaka Yoshida [hirotaka.yoshida.qv@hitachi.com]
Sent: Friday, June 26, 2009 9:24 AM
To: Multiple recipients of list
Subject: OFFICIAL COMMENT: Lesamnta

Attachments: Security_Analysis_Compression_Lesamnta.pdf

Security_Analysis_
Compression_...

Dear NIST, all,

We send a report on a security analysis of the compression function of Lesamnta. In this report, we have discussed the security analysis of the compression function of Lesamnta that was pointed by Charles Bouillaguet, Orr Dunkelman, Gaetan Leurent, Pierre-Alain Fouque. As the result of examining several attacking scenarios based on this analysis, we conclude that the expected strength of Lesamnta described still remains the same despite of the loss of proved security regarding preimage resistance, second preimage resistance, and collision resistance.
In order for Lesamnta to get back proved security on each of these security requirements, we will make a minor change to the specification by changing round constants.

Best regards,
Hirotaka Yoshida

Security Analysis of the Compression Function of Lesamnta and its Impact

Shoichi HIROSE[1], Hidenori KUWAKADO[2], Hirotaka YOSHIDA[3,4]

[1] University of Fukui
hrs_shch@u-fukui.ac.jp
[2] Kobe University
kuwakado@kobe-u.ac.jp
[3] Systems Development Laboratory, Hitachi, Ltd.,
hirotaka.yoshida.qv@hitachi.com
[4] Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,

1 Introduction

Lesamnta is a new family of hash functions submitted to NIST for their cryptographic hash algorithm competition.

A security analysis of the compression function of Lesamnta has been reported [1]. In this document, we give a short overview of how this analysis affects the security of the full Lesamnta hash function. We divide our arguments into three categories:

- A security analysis of the Lesamnta compression function
- The impact of the security analysis on the security of the full Lesamnta
- A plan for a minor change to the specification

2 A Security Analysis of the Compression Function

2.1 Observation on Lesamnta's Block Cipher

This section describes a correlation among a key, a plaintext, and a ciphertext in Lesamnta's block cipher. The correlation was discovered by Bouillaguet *et al.* [1]. We only describe the observation on Lesamnta-256, but we can obtain similar observation on Lesamnta-512; the difference is just word size.

We follow symbols and notations of [2] and consider Lesamnta-256's block cipher $EncComp_{256}$. Let $\mathtt{C}[r][0]$ and $\mathtt{C}[r][1]$ be the left part and the right part of the r-th round constant in the key schedule function (see Figure 18 of [2]). For example, $\mathtt{C}[0][0] = \mathtt{00000001}$ and $\mathtt{C}[0][1] = \mathtt{00000000}$ according to p.14 of [2]. We define a difference Δ_r as

$$\Delta_r = \mathtt{C}[r][0] \oplus \mathtt{C}[r][1] \tag{1}$$

for $r = 0, 1, \ldots, 31$. According to p.14 of [2], we see that the following equations hold.

$$\Delta_0 = \Delta_4 = \Delta_8 = \Delta_{12} = \ldots = \Delta_{24} = \Delta_{28},$$

$$\begin{aligned}
&\Delta_1 = \Delta_5 = \Delta_9 = \Delta_{13} = \ldots = \Delta_{25} = \Delta_{29}, \\
&\Delta_2 = \Delta_6 = \Delta_{10} = \Delta_{14} = \ldots = \Delta_{26} = \Delta_{30}, \\
&\Delta_3 = \Delta_7 = \Delta_{11} = \Delta_{15} = \ldots = \Delta_{27}.
\end{aligned} \tag{2}$$

Precisely speaking, we also see $\Delta_0 = \Delta_1 = \Delta_2 = \Delta_3 = \mathsf{00000001}$ and $\Delta_3 = \Delta_{31}$, but these properties are unnecessary for the following discussion. We will see that the relation of Eq. (2) allows an adversary to attack Lesamnta.

A key `chain`, which corresponds to `chain`[8] in Figure 18 of [2], is denoted by $\mathtt{chain}[0] \parallel \mathtt{chain}[1] \parallel \ldots \parallel \mathtt{chain}[7]$ where $\mathtt{chain}[i] \in \{0,1\}^{32}$. The key schedule function produces 32 round keys $\mathtt{K}[0][0]||\mathtt{K}[0][1], \ldots, \mathtt{K}[31][0]||\mathtt{K}[31][1]$ from the key.

Proposition 1. *Let* $\mathtt{chain}_0$ *be any key* $\mathtt{chain}_0[0] \parallel \mathtt{chain}_0[1] \parallel \ldots \parallel \mathtt{chain}_0[7]$. *Suppose that another key* $\mathtt{chain}_1$ *is determined as*

$$\begin{aligned}
\mathtt{chain}_1 = &(\mathtt{chain}_0[1] \oplus \Delta_2) \parallel (\mathtt{chain}_0[0] \oplus \Delta_2) \parallel (\mathtt{chain}_0[3] \oplus \Delta_1) \parallel (\mathtt{chain}_0[2] \oplus \Delta_1) \\
&\parallel (\mathtt{chain}_0[5] \oplus \Delta_0) \parallel (\mathtt{chain}_0[4] \oplus \Delta_0) \parallel (\mathtt{chain}_0[7] \oplus \Delta_3) \parallel (\mathtt{chain}_0[6] \oplus \Delta_3).
\end{aligned}$$

When round keys $\mathtt{K}_0$ *generated from* $\mathtt{chain}_0$ *are denoted by*

$$\mathtt{K}_0[0][0]||\mathtt{K}_0[0][1], \mathtt{K}_0[1][0]||\mathtt{K}_0[1][1], \ldots, \mathtt{K}_0[31][0]||\mathtt{K}_0[31][1],$$

round keys $\mathtt{K}_1$ *generated from* $\mathtt{chain}_1$ *are given by*

$$\begin{aligned}
&\mathtt{K}_1[4i][0] = \mathtt{K}_0[4i][1] \oplus \Delta_2, && \mathtt{K}_1[4i][1] = \mathtt{K}_0[4i][0] \oplus \Delta_2, \\
&\mathtt{K}_1[4i+1][0] = \mathtt{K}_0[4i+1][1] \oplus \Delta_3, && \mathtt{K}_1[4i+1][1] = \mathtt{K}_0[4i+1][0] \oplus \Delta_3, \\
&\mathtt{K}_1[4i+2][0] = \mathtt{K}_0[4i+2][1] \oplus \Delta_0, && \mathtt{K}_1[4i+2][1] = \mathtt{K}_0[4i+2][0] \oplus \Delta_0, \\
&\mathtt{K}_1[4i+3][0] = \mathtt{K}_0[4i+3][1] \oplus \Delta_1, && \mathtt{K}_1[4i+3][1] = \mathtt{K}_0[4i+3][0] \oplus \Delta_1,
\end{aligned}$$

for $i = 0, 1, \ldots, 7$.

Next, consider the mixing function of the block cipher $EncComp_{256}$ (see Figure 11 of [2]). A message block `mb` is denoted by $\mathtt{mb}[0] \parallel \mathtt{mb}[1] \parallel \ldots \parallel \mathtt{mb}[7]$ where $\mathtt{mb}[i] \in \{0,1\}^{32}$. Let $\mathtt{K}[0][0]||\mathtt{K}[0][1], \ldots, \mathtt{K}[31][0]||\mathtt{K}[31][1]$ be 32 round keys generated by the key schedule function. The output `x` of $EncComp_{256}$ (i.e., the ciphertext) is denoted by $\mathtt{x}[0] \parallel \mathtt{x}[1] \parallel \ldots \parallel \mathtt{x}[7]$.

Proposition 2. *Let* $\mathtt{K}_0[0][0]||\mathtt{K}_0[0][1], \ldots, \mathtt{K}_0[31][0]||\mathtt{K}_0[31][1]$ *be 32 round keys* $\mathtt{K}_0$, *and let* $\mathtt{mb}_0[0] \parallel \mathtt{mb}_0[1] \parallel \ldots \parallel \mathtt{mb}_0[7]$ *denote a message block* $\mathtt{mb}_0$. *Suppose that round keys* $\mathtt{K}_1$ *and a message block* $\mathtt{mb}_1$ *satisfy the following equations.*

$$\begin{aligned}
&\mathtt{K}_1[4i][0] = \mathtt{K}_0[4i][1] \oplus \delta_0, && \mathtt{K}_1[4i][1] = \mathtt{K}_0[4i][0] \oplus \delta_0, \\
&\mathtt{K}_1[4i+1][0] = \mathtt{K}_0[4i+1][1] \oplus \delta_1, && \mathtt{K}_1[4i+1][1] = \mathtt{K}_0[4i+1][0] \oplus \delta_1, \\
&\mathtt{K}_1[4i+2][0] = \mathtt{K}_0[4i+2][1] \oplus \delta_2, && \mathtt{K}_1[4i+2][1] = \mathtt{K}_0[4i+2][0] \oplus \delta_2, \\
&\mathtt{K}_1[4i+3][0] = \mathtt{K}_0[4i+3][1] \oplus \delta_3, && \mathtt{K}_1[4i+3][1] = \mathtt{K}_0[4i+3][0] \oplus \delta_3, \\
&\mathtt{mb}_1[0] = \mathtt{mb}_0[1] \oplus \delta_2, && \mathtt{mb}_1[1] = \mathtt{mb}_0[0] \oplus \delta_2, \\
&\mathtt{mb}_1[2] = \mathtt{mb}_0[3] \oplus \delta_1, && \mathtt{mb}_1[3] = \mathtt{mb}_0[2] \oplus \delta_1, \\
&\mathtt{mb}_1[4] = \mathtt{mb}_0[5] \oplus \delta_0, && \mathtt{mb}_1[5] = \mathtt{mb}_0[4] \oplus \delta_0, \\
&\mathtt{mb}_1[6] = \mathtt{mb}_0[7] \oplus \delta_3, && \mathtt{mb}_1[7] = \mathtt{mb}_0[6] \oplus \delta_3,
\end{aligned}$$

where $i = 0, 1, \ldots, 7$ *and* $\delta_0, \ldots, \delta_3$ *are any 32-bit strings. Let* $\mathtt{x}_0[0] \parallel \mathtt{x}_0[1] \parallel \ldots \parallel \mathtt{x}_0[7]$ *be the output* $\mathtt{x}_0$ *of* $EncComp_{256}(\mathtt{K}_0, \mathtt{mb}_0)$*. Then, the output* $\mathtt{x}_1$ *of* $EncComp_{256}(\mathtt{K}_1, \mathtt{mb}_1)$ *is given by*

$$\begin{array}{ll} \mathtt{x}_1[0] = \mathtt{x}_0[1] \oplus \delta_2, & \mathtt{x}_1[1] = \mathtt{x}_0[0] \oplus \delta_2, \\ \mathtt{x}_1[2] = \mathtt{x}_0[3] \oplus \delta_1, & \mathtt{x}_1[3] = \mathtt{x}_0[2] \oplus \delta_1, \\ \mathtt{x}_1[4] = \mathtt{x}_0[5] \oplus \delta_0, & \mathtt{x}_1[5] = \mathtt{x}_0[4] \oplus \delta_0, \\ \mathtt{x}_1[6] = \mathtt{x}_0[7] \oplus \delta_3, & \mathtt{x}_1[7] = \mathtt{x}_0[6] \oplus \delta_3. \end{array}$$

Proposition 1 and Proposition 2 are proved by using properties of internal functions such as `SubWord256`. Assuming that

$$\delta_0 = \Delta_2,\ \delta_1 = \Delta_3,\ \delta_2 = \Delta_0,\ \delta_3 = \Delta_1,$$

we obtain the following proposition from Proposition 1 and Proposition 2.

Proposition 3. *Let* $\mathtt{chain}_0$ *and* $\mathtt{mb}_0$ *be a key and a message block, respectively.*

$$\begin{aligned} \mathtt{chain}_0 = \mathtt{chain}_0[0] \parallel \mathtt{chain}_0[1] \parallel \mathtt{chain}_0[2] \parallel \mathtt{chain}_0[3] \\ \parallel \mathtt{chain}_0[4] \parallel \mathtt{chain}_0[5] \parallel \mathtt{chain}_0[6] \parallel \mathtt{chain}_0[7], \\ \mathtt{mb}_0 = \mathtt{mb}_0[0] \parallel \mathtt{mb}_0[1] \parallel \mathtt{mb}_0[2] \parallel \mathtt{mb}_0[3] \\ \parallel \mathtt{mb}_0[4] \parallel \mathtt{mb}_0[5] \parallel \mathtt{mb}_0[6] \parallel \mathtt{mb}_0[7]. \end{aligned}$$

Suppose that a key $\mathtt{chain}_1$ *and a message block* $\mathtt{mb}_1$ *are given as*

$$\begin{aligned} \mathtt{chain}_1 = (\mathtt{chain}_0[1] \oplus \Delta_2) \parallel (\mathtt{chain}_0[0] \oplus \Delta_2) \\ \parallel (\mathtt{chain}_0[3] \oplus \Delta_1) \parallel (\mathtt{chain}_0[2] \oplus \Delta_1) \\ \parallel (\mathtt{chain}_0[5] \oplus \Delta_0) \parallel (\mathtt{chain}_0[4] \oplus \Delta_0) \\ \parallel (\mathtt{chain}_0[7] \oplus \Delta_3) \parallel (\mathtt{chain}_0[6] \oplus \Delta_3), \end{aligned} \tag{3}$$

$$\begin{aligned} \mathtt{mb}_1 = (\mathtt{mb}_0[1] \oplus \Delta_0) \parallel (\mathtt{mb}_0[0] \oplus \Delta_0) \parallel (\mathtt{mb}_0[3] \oplus \Delta_3) \parallel (\mathtt{mb}_0[2] \oplus \Delta_3) \\ \parallel (\mathtt{mb}_0[5] \oplus \Delta_2) \parallel (\mathtt{mb}_0[4] \oplus \Delta_2) \parallel (\mathtt{mb}_0[7] \oplus \Delta_1) \parallel (\mathtt{mb}_0[6] \oplus \Delta_1). \end{aligned} \tag{4}$$

When the output $\mathtt{x}_0$ *of* $EncComp_{256}(\mathtt{chain}_0, \mathtt{mb}_0)$ *is denoted by* $\mathtt{x}_0[0] \parallel \mathtt{x}_0[1] \parallel \ldots \parallel \mathtt{x}_0[7]$*, the output* $\mathtt{x}_1$ *of* $EncComp_{256}(\mathtt{chain}_1, \mathtt{mb}_1)$ *is given by*

$$\begin{aligned} \mathtt{x}_1 = (\mathtt{x}_0[1] \oplus \Delta_0) \parallel (\mathtt{x}_0[0] \oplus \Delta_0) \parallel (\mathtt{x}_0[3] \oplus \Delta_3) \parallel (\mathtt{x}_0[2] \oplus \Delta_3) \\ \parallel (\mathtt{x}_0[5] \oplus \Delta_2) \parallel (\mathtt{x}_0[4] \oplus \Delta_2) \parallel (\mathtt{x}_0[7] \oplus \Delta_1) \parallel (\mathtt{x}_0[6] \oplus \Delta_1). \end{aligned} \tag{5}$$

2.2 Distinguisher for Lesamnta's Block Cipher

Proposition 3 immediately gives an efficient related-key adversary A for distinguishing between Lesamnta-256's block cipher $EncComp_{256}$ and an ideal cipher IC. The basic idea of this distinguisher was shown in [1].

The algorithm of the adversary A is described below. Suppose that a block cipher BC to which A has access is promised to be either $EncComp_{256}$ or IC and A is allowed to have access to the related-key oracle such as Eq. (3). Namely, A does not know keys $\mathtt{chain}_0, \mathtt{chain}_1$, but A can have access to $BC(\mathtt{chain}_0, \cdot)$ and $BC(\mathtt{chain}_1, \cdot)$.

1. Choose a message block $\mathtt{mb}_0$ at random and determine another message block $\mathtt{mb}_1$ as Eq. (4).
2. Let $\mathtt{x}_i$ be the output of $BC(\mathtt{chain}_i, \mathtt{mb}_i)$ where $i = 0, 1$. If Eq. (5) holds, then output 1, otherwise output 0.

We evaluate the probability that A outputs 1. If BC is $EncComp_{256}$, then A always outputs 1 because of Proposition 3. If BC is IC, then the probability that A outputs 1 is 2^{-256}. Thus, A can distinguish between Lesamnta-256's block cipher and the ideal cipher by making only two queries.

2.3 Pseudo-Collision of Lesamnta

The sophisticate use of Proposition 3 allows an adversary to produce a pseudo-collision of Lesamnta-256 with $O(2^{64})$ computations of the compression function. This attack was shown in [1].

Consider Lesamnta-256's compression function `Compression256` (Figure 11 of [2]). The algorithm of an adversary A that finds a pseudo-collision is described below.

1. Let a set $\mathcal{U} = \emptyset$.
2. For $i = 1, 2, \ldots, 2^{64}$, do the following steps.
   2.1 Choose $\mathtt{chain}_i[j], \mathtt{mb}_i[j]$ where $j = 0, 2, 4, 6$ at random.
   2.2 Determine $\mathtt{chain}_i, \mathtt{mb}_i$ as follows:

$$\begin{aligned} \mathtt{chain}_i = {} & \mathtt{chain}_i[0] \parallel (\mathtt{chain}_i[0] \oplus \Delta_2) \\ & \parallel \mathtt{chain}_i[2] \parallel (\mathtt{chain}_i[2] \oplus \Delta_1) \\ & \parallel \mathtt{chain}_i[4] \parallel (\mathtt{chain}_i[4] \oplus \Delta_0) \\ & \parallel \mathtt{chain}_i[6] \parallel (\mathtt{chain}_i[6] \oplus \Delta_3) \end{aligned} \tag{6}$$

$$\begin{aligned} \mathtt{mb}_i = {} & \mathtt{mb}_i[0] \parallel (\mathtt{mb}_i[0] \oplus \Delta_0) \parallel \mathtt{mb}_i[2] \parallel (\mathtt{mb}_i[2] \oplus \Delta_3) \\ & \parallel \mathtt{mb}_i[4] \parallel (\mathtt{mb}_i[4] \oplus \Delta_2) \parallel \mathtt{mb}_i[6] \parallel (\mathtt{mb}_i[6] \oplus \Delta_1) \end{aligned} \tag{7}$$

   2.3 Compute $\mathtt{Compression256}(\mathtt{chain}_i, \mathtt{mb}_i)$. The output is denoted by $\mathtt{z}_i$.
   2.4 Let $\mathcal{U} \leftarrow \mathcal{U} \cup (\mathtt{chain}_i, \mathtt{mb}_i, \mathtt{z}_i)$.
3. Find $(\mathtt{chain}_\iota, \mathtt{mb}_\iota)$ and $(\mathtt{chain}_\nu, \mathtt{mb}_\nu)$ such that $\mathtt{z}_\iota = \mathtt{z}_\nu$ from $\mathcal{U}$. (i.e., a pseudo-collision).

Recall that Lesamnta's compression functions is the MMO mode. The output $\mathtt{z}_i$ of $\mathtt{Compression256}(\mathtt{chain}_i, \mathtt{mb}_i)$ always satisfies the following property due to Proposition 3.

$$\mathtt{z}_i[0] = \mathtt{z}_i[1],\ \mathtt{z}_i[2] = \mathtt{z}_i[3],\ \mathtt{z}_i[4] = \mathtt{z}_i[5],\ \mathtt{z}_i[6] = \mathtt{z}_i[7].$$

Namely, the size of the output space of $\mathtt{Compression256}(\mathtt{chain}_i, \mathtt{mb}_i)$ is 2^{128}. Since $\mathcal{U}$ has 2^{64} elements, there exists a pair satisfying step 3 with probability $1 - 1/\mathrm{e}$ due to the birthday paradox.

3 The Impact of the Security Analysis of the Compression Function on the Full Lesamnta

In this section, we discuss the impact of the security analysis described in 2 on the security of the full Lesamnta by firstly reviewing the expected strength and security goals claimed in [2] and by secondly considering several attacking scenarios.

3.1 Review of What Was Claimed in [2]

In the section of "Expected Strength and Security Goals" in [2], we described as follows:

Table 1 shows the expected strength of Lesamnta for each of the security requirements (i.e., the expected complexity of attacks). What values in Table 1 mean is explained below. The row indicated by "HMAC" lists the approximate number of queries required by any distinguishing attack against HMAC using Lesamnta. The row indicated by "PRF" lists the approximate number of queries required by any distinguishing attack against the additional PRF modes described in Sec. 13.1. The row indicated by "Randomized hashing" lists the approximate complexity to find another pair of a message and a random value for a given pair of a 2^k-bit message and a random value. The fourth row lists the approximate complexity of any collision attack. The fifth row lists the approximate complexity of any preimage attack. The sixth row lists the approximate complexity of the Kelsey-Schneier second-preimage attack with any first preimage shorter than 2^k bits. The seventh row lists the approximate number of queries required by any length-extension attack against Lesamnta. A cryptanalytic attack may be a profound threat to Lesamnta if its complexity is much less than the complexity in Table 1.

Table 1. Expected strength of Lesamnta

Requirement	Lesamnta			
	224	256	384	512
HMAC	2^{112}	2^{128}	2^{192}	2^{256}
PRF	2^{112}	2^{128}	2^{192}	2^{256}
Randomized hashing	2^{256-k}	2^{256-k}	2^{512-k}	2^{512-k}
Collision resistance	2^{112}	2^{128}	2^{192}	2^{256}
Preimage resistance	2^{224}	2^{256}	2^{384}	2^{512}
Second-preimage resistance	2^{256-k}	2^{256-k}	2^{512-k}	2^{512-k}
Length-extension attacks	2^{112}	2^{128}	2^{192}	2^{256}

Table 1 includes proof-based strength and attack-based strength. The security proof of Lesamnta is given as follows:

Proved security 1: *Lesamnta is indifferentiable from a random oracle under the assumption that block ciphers E, L are independent ideal ciphers.*
This proof partially ensures the security of randomized hashing, collision resistance, preimage resistance, second-preimage resistance, and length-extension attacks.

Proved security 2: *Lesamnta is collision resistant under the assumption that the compression function h and the output function g are collision resistant.*
This proof ensures the security of collision resistance, and in part, preimage resistance and second-preimage resistance.

Proved security 3: *Lesamnta is a pseudorandom function under the assumption that block ciphers E, L are independent pseudorandom permutations.*
This proof ensures the security of HMAC and PRF.

We claim that the impact of the security analysis of the compression function on the security of Lesamnta described in 2 is limited to the following:

– Each of the assumption made in Proved Security 1 and the one in Proved Security 2 no longer holds because the above attack means that Lesamnta's block cipher is a *poor* instantiation of an ideal cipher.

We claim that there is no problem regarding Proved Security 3 because their proofs only assume the pseudo-randomness of the underlying block ciphers, that is, the key is secret and chosen at random.

3.2 Collision Resistance, Second-preimage Resistance, and Preimage Resistance

As for collision resistance, second-preimage resistance, and preimage resistance, Lesamnta does not have proof-based strength but we still claim that, regarding each of these security requirements, Lesamnta has attack-based strength which is estimated in security analysis described in [2] together with the arguments we describe below.

As for collision resistance and second-preimage resistance, we think that it is difficult to transform the collision attack on the compression function given in Section 2 into an attack on the full Lesamnta hash function because it is not clear how to find the chaining variable H_i of the specific form described in Section 2 for the full Lesamnta.

As for preimage resistance, we do not know any way to transform the pseudo-collision attack given in Section 2 into a preimage attack on the full Lesamnta.

3.3 Security against a Collision Attack on the Full Lesamnta

Using Proposition 3, we can find a collision of Lesamnta hash function with the same complexity of a generic attack.

Consider Lesamnta-256. The algorithm of an adversary that finds a collision is described below.

1. Let $\mathcal{U}^{(0)} = \emptyset$ and $\mathcal{U}^{(1)} = \emptyset$.
2. Choose message block blocks $\mathtt{mb}_i^{(0)}, \mathtt{mb}_i^{(1)}$ at random. If $\mathtt{mb}_i^{(1)}$ satisfies the following equations (i.e., Eq. (7)), then choose $\mathtt{mb}_i^{(1)}$ again.

$$\begin{aligned} \mathtt{mb}_i^{(1)}[1] = \mathtt{mb}_i^{(1)}[0] \oplus \Delta_0, \quad \mathtt{mb}_i^{(1)}[3] = \mathtt{mb}_i^{(1)}[2] \oplus \Delta_3, \\ \mathtt{mb}_i^{(1)}[5] = \mathtt{mb}_i^{(1)}[4] \oplus \Delta_2, \quad \mathtt{mb}_i^{(1)}[7] = \mathtt{mb}_i^{(1)}[6] \oplus \Delta_1, \end{aligned}$$

   where Δ_i is given by Eq. (1).
3. Compute

$$\begin{aligned} \mathtt{chain}_i^{(0)} &= \mathtt{Compression256}(IV, \mathtt{mb}_i^{(0)}), \\ \mathtt{chain}_i^{(1)} &= \mathtt{Compression256}(\mathtt{chain}_i^{(0)}, \mathtt{mb}_i^{(1)}), \end{aligned}$$

   where IV is the standard initial value (Section 5.2.3.2 of [2]).
4. Let $\mathcal{U}^{(0)} \leftarrow \mathcal{U}^{(0)} \cup (\mathtt{chain}_i^{(0)}, \mathtt{mb}_i^{(0)})$ and $\mathcal{U}^{(1)} \leftarrow \mathcal{U}^{(1)} \cup (\mathtt{chain}_i^{(1)}, \mathtt{mb}_i^{(1)})$.
5. If all the following conditions hold, then go to the next step, otherwise go back to step 2.
   - There is an element in $\mathcal{U}^{(0)}$ satisfying Eq. (6), that is, for some i

$$\begin{aligned} \mathtt{chain}_i^{(0)}[1] = \mathtt{chain}_i^{(0)}[0] \oplus \Delta_2, \quad \mathtt{chain}_i^{(0)}[3] = \mathtt{chain}_i^{(0)}[2] \oplus \Delta_1, \\ \mathtt{chain}_i^{(0)}[5] = \mathtt{chain}_i^{(0)}[4] \oplus \Delta_0, \quad \mathtt{chain}_i^{(0)}[7] = \mathtt{chain}_i^{(0)}[6] \oplus \Delta_2. \end{aligned}$$

     This index i is denoted by i_0.
   - There is an element in $\mathcal{U}^{(1)}$ such that for some i

$$\mathtt{chain}_i^{(1)}[j] = \mathtt{chain}_i^{(1)}[j+1]$$

     for $j = 0, 2, 4, 6$. This index i is denoted by i_1.
6. Choose a message block $\mathtt{mb}'^{(1)}$ at random such that

$$\begin{aligned} \mathtt{mb}'^{(1)}[1] = \mathtt{mb}'^{(1)}[0] \oplus \Delta_0, \quad \mathtt{mb}'^{(1)}[3] = \mathtt{mb}'^{(1)}[2] \oplus \Delta_3, \\ \mathtt{mb}'^{(1)}[5] = \mathtt{mb}'^{(1)}[4] \oplus \Delta_2, \quad \mathtt{mb}'^{(1)}[7] = \mathtt{mb}'^{(1)}[6] \oplus \Delta_1. \end{aligned}$$

7. Compute

$$\mathtt{chain}'^{(1)} = \mathtt{Compression256}(\mathtt{chain}_{i_0}^{(0)}, \mathtt{mb}'^{(1)}).$$

8. If the following equations hold, then output $\mathtt{mb}_{i_1}^{(0)} \parallel \mathtt{mb}_{i_1}^{(1)}$ and $\mathtt{mb}_{i_0}^{(0)} \parallel \mathtt{mb}'^{(1)}$ as a collision-message pair, that is,

$$\mathtt{chain}'^{(1)}[j] = \mathtt{chain}_{i_1}^{(1)}[j],$$

   for $j = 0, 1, \ldots, 7$. Otherwise go back to step 6.

We evaluate the complexity of the above algorithm. In order to satisfy the conditions in step 5 and the condition in step 8, $O(2^{128})$ computations of the compression function are required. As a result, we conclude that the above attack is not better than the generic collision attack. This means that this attack does not pose any threat on the full Lesamnta.

4 A Plan for a Minor Change

We observe that the security analysis discussed here is based on some symmetry in Lesamnta. To destroy the symmetry, we plan to make a minor change to the specification of Lesamnta by changing the round constants. The important design goals for the new round constants are security and hardware efficiency.

The possible ideas for new round constants are using the following techniques: LFSR, publicly known random-looking numbers, pseudo-random generators, etc. We also consider the possibility of using the on-the-fly technique and the adaptability to the extension of Lesamnta specified in [2].

5 Concluding Remarks

In this paper, we have discussed the security analysis of the compression function of Lesamnta that was pointed by Bouillaguet *et al.* As the result of examining several attacking scenarios based on this analysis, we conclude that the expected strength of Lesamnta described still remains the same despite of the loss of proved security regarding preimage resistance, second preimage resistance, and collision resistance.

In order for Lesamnta to get back proved security on each of these security requirements, we will make a minor change to the specification by changing round constants.

Acknowledgments

We would like to thank Charles Bouillaguet, Orr Dunkelman, Gaëtan Leurent, Pierre-Alain Fouque for their excellent analysis on Lesamnta. We also thank Kota Ideguchi, Yasuko Fukuzawa, and Toru Owada for fruitful discussions. This work was partially supported by the National Institute of Information and Communications Technology, Japan.

References

1. C. Bouillaguet, O. Dunkelman, G. Leurent, and P. A. Fouque, Private communication, 2009.
2. S. Hirose, H. Kuwakado, and H. Yoshida, "SHA-3 proposal: Lesamnta," `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/Lesamnta.zip%`, October 2008. latest version: `http://www.sdl.hitachi.co.jp/crypto/lesamnta/`.
3. National Institute of Standards and Technology, "Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family," `http://csrc.nist.gov/groups/ST/hash/documents/`, November 2007.

From: hash-forum@nist.gov on behalf of Gilles VAN ASSCHE [gilles.vanassche@st.com]
Sent: Monday, July 06, 2009 8:35 AM
To: Multiple recipients of list
Subject: RAM usage (Re: OFFICIAL COMMENT: Lesamnta)

Dear Hirotaka Yoshida,

> We compare the implementation costs of various SHA-3 candidates on
> low-cost 8-bit CPUs by estimating RAM/ROM requirements of them.

In your document, you assume that the message block is counted towards the memory usage of the application. It is a valid assumption in several cases. However, there are also applications for which the message is formatted on the fly or does not need to be kept after being hashed.
There, constructions such as sponge functions or similar (e.g., CubeHash, LUX) can directly XOR the message block into the state, relieving the application from dedicating a memory area for it. This optimization also applies where the hashing API is composed of functions such as Init, Update and Final. In general a message queue must be allocated, which can be avoided for sponge functions or similar.

About Keccak specifically, the designer of an application on a memory-constrained device may also opt for a smaller state size by using an alternate set of parameters, such as Keccak[r=288,c=512], which uses 100 bytes of RAM. And if 256 bits of capacity are enough for such an application, Keccak[r=144,c=256] uses only 50 bytes.

Kind regards,
The Keccak team

From: hash-forum@NIST.GOV on behalf of Hirotaka Yoshida [hirotaka.yoshida.qv@hitachi.com]
Sent: Wednesday, July 15, 2009 6:18 AM
To: Multiple recipients of list
Subject: Re: RAM usage (Re: OFFICIAL COMMENT: Lesamnta)

Dear The Keccak team,

Thank you very much for your interest and your comments on our document.

We understand that the ways of counting the message block depend on
the applications. We agree with you that there are also applications for which the message is formatted on the fly or does not need to be kept after being hashed.

Our document compares the implementation costs of candidates under a simple but valid assumption as the first step in this kind of research.
However, we think that it is interesting to compare them under the assumption you suggested as well. We hope that other groups investigate in this direction.

Best regards,
Hirotaka Yoshida

> Dear Hirotaka Yoshida,
>
>> We compare the implementation costs of various SHA-3 candidates on
>> low-cost 8-bit CPUs by estimating RAM/ROM requirements of them.
>
> In your document, you assume that the message block is counted towards
> the memory usage of the application. It is a valid assumption in
> several cases. However, there are also applications for which the
> message is formatted on the fly or does not need to be kept after being hashed.
> There, constructions such as sponge functions or similar (e.g.,
> CubeHash, LUX) can directly XOR the message block into the state,
> relieving the application from dedicating a memory area for it. This
> optimization also applies where the hashing API is composed of
> functions such as Init, Update and Final. In general a message queue
> must be allocated, which can be avoided for sponge functions or similar.
>
> About Keccak specifically, the designer of an application on a
> memory-constrained device may also opt for a smaller state size by
> using an alternate set of parameters, such as Keccak[r=288,c=512],
> which uses 100 bytes of RAM. And if 256 bits of capacity are enough
> for such an application, Keccak[r=144,c=256] uses only 50 bytes.
>
> Kind regards,
> The Keccak team
>
>

From: Nandi, Mridul [mridul.nandi@nist.gov]
Sent: Wednesday, July 15, 2009 11:53 PM
To: internal-hash@nist.gov
Subject: FW: OFFICIAL COMMENT: Lesamnta

Hi All,
I forgot to attach a reply from Lesamnta Designer I received last week regarding my observation on Lesamnta.
-Mridul

_____

From: Hirotaka Yoshida [hirotaka.yoshida.qv@hitachi.com]
Sent: Thursday, July 09, 2009 7:27 PM
To: Nandi, Mridul
Cc: hirotaka.yoshida.qv@hitachi.com; hrs_shch@u-fukui.ac.jp; kuwakado@kobe-u.ac.jp
Subject: RE: FW: OFFICIAL COMMENT: Lesamnta

Dear Dr. Mridul Nandi,

Thank you for your comments on our paper on the security of Lesamnta.

> 1. Semi-free start collision in 2^{64} complexity. The same attack as
> described in the paper by Charles Bouillaguet, Orr Dunkelman, Gaetan
> Leurent, Pierre-Alain Fouque where the chaining value keeps fixed. I
> do not find any reason to vary chaining value. Moreover,
> E_{chain}(mb) + mb is not expected to be an one-one function. So we
> should get collision after 2^{64} tries of mb. Please point me if I'm
> wrong or missing something.

Your semi-free start attack works for Lesamnta in 2^{64} complexity.
Actually, we were trying to present the attack in as general form as possible. So, the special (but important) attack, semi-free start attack, is not explicit in the presentation.

> 2. Moreover, one can have preimage attack on Lesamnta-256 (without the
> length padding) in 2^{128} complexity for any targets of the form
> z_0|| z_0 || z_2 || z_2 || z_4 || z_4 || z_6 || z_6. This can be done
> for two blocks. We first vary 1st message block to reach the
> intermediate chaining value of the same form as in the previous attack
> (mentioned in the paper) and then we choose the message in a
> particular form to get the specific preimage. However, the 10^{191}
> || length does not satisfy this specific pattern and this is why I
> do not know how it can be applied with the padding.

We agree with you that the preimage attack you described in the above can be applied to the Lesamnta-256 without the length padding. We expect that this preimage attack cannot be applied to the full Lesamnta.

As we described in the paper, we will make a minor change to the specification by changing round constants.
This change of round constants prevents the above attacks.
We will send you (NIST ML) a report describing this change in weeks.

Sincerely,
Lesamnta Design Team

>Dear Lesamnta Designers,
>

>After going through the Proposition 3 of the attached paper, I can see the following observations:
>
>1. Semi-free start collision in 2^{64} complexity. The same attack as
>described in the paper by Charles Bouillaguet, Orr Dunkelman, Gaetan Leurent, Pierre-Alain Fouque where the chaining value keeps fixed. I do not find any reason to vary chaining value. Moreover, E_{chain}(mb) + mb is not expected to be an one-one function. So we should get collision after 2^{64} tries of mb. Please point me if I'm wrong or missing something.
>
>
>2. Moreover, one can have preimage attack on Lesamnta-256 (without the
>length padding) in 2^{128} complexity for any targets of the form
>z_0|| z_0 || z_2 || z_2 || z_4 || z_4 || z_6 || z_6. This can be done
>for two blocks. We first vary 1st message block to reach the
>intermediate chaining value of the same form as in the previous attack
>(mentioned in the paper) and then we choose the message in a particular
>form to get the specific preimage. However, the 10^{191} || length does
>not satisfy this specific p
attern and this is why I do not know how it can be applied with the padding.
>
>Please feel free to share what you think on these attacks.
>
>Thanks and regards,
>Mridul
>
>
>-----Original Message-----
>From: hash-forum@nist.gov [mailto:hash-forum@nist.gov] On Behalf Of
>Hirotaka Yoshida
>Sent: Friday, June 26, 2009 9:24 AM
>To: Multiple recipients of list
>Subject: OFFICIAL COMMENT: Lesamnta
>
>Dear NIST, all,
>
>We send a report on a security analysis of the compression function of
>Lesamnta. In this report, we have discussed the security analysis of
>the compression function of Lesamnta that was pointed by Charles
>Bouillaguet, Orr Dunkelman, Gaetan Leurent, Pierre-Alain Fouque. As the
>result of examining several attacking scenarios based on this
>analysis, we conclude that the expected strength of Lesamnta described
>still remains the same despite of the loss of proved security regarding
>preimage resistance, second preimage resistance, and collision
>resistance.
>In order for Lesamnta to get back proved security on each of these
>security requirements, we will make a minor change to the specification
>by changing round constants.
>
>Best regards,
>Hirotaka Yoshida
>

From: hash-forum@nist.gov on behalf of Hirotaka Yoshida [hirotaka.yoshida.qv@hitachi.com]
Sent: Friday, July 17, 2009 1:33 PM
To: Multiple recipients of list
Subject: OFFICIAL COMMENT: Lesamnta

Attachments: Minor_Change_Lesamnta.pdf

Minor_Change_Les amnta.pdf (152...

Dear NIST, all,

We would like to send you a document where we propose to make a minor change to the specification of Lesamnta by changing round constants.
The motivation of this change is to prevent the attacks
described in [1] and the newly-discovered attack described in this document.
We expect that all of these attacks do not work any more for Lesamnta with the new round constants. We also expect that the change does not cause any significant impact on resistance against the known attacks and on performance of Lesamnta.

Reference:
[1] S. Hirose, H. Kuwakado, H. Yoshida, "Security Analysis of the Compression Function of Lesamnta and its Impact"
http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/LESAMNTA_Comments.pdf

Best regards,
The Lesamnta design team

A Minor Change to Lesamnta — Change of Round Constants —

Shoichi Hirose[1], Hidenori Kuwakado[2], Hirotaka Yoshida[3,4]

[1] University of Fukui
hrs_shch@u-fukui.ac.jp
[2] Kobe University
kuwakado@kobe-u.ac.jp
[3] Systems Development Laboratory, Hitachi, Ltd.,
hirotaka.yoshida.qv@hitachi.com
[4] Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC

1 Introduction

Lesamnta is a new family of hash functions submitted to NIST for their cryptographic hash algorithm competition.

In this document, we propose to make a minor change to the specification of Lesamnta by changing round constants. We give a short overview of the minor change to the Lesamnta hash function. We divide our arguments into the following categories:

- The minor change of the Lesamnta hash function
- The motivation of the change
- The design principle for the round constants
- Security against potential attacks on Lesamnta with new round constants
- The impact of the change

Note that we mainly explain about the minor change in the case of Lesamnta-256 because the explanation could be easily adapted to the case of Lesamnta-224/384/512.

2 The Minor Change

We propose to make a minor change to the specification of Lesamnta [2]. The minor change is only the replacement of 32 round constats. No other parts of the specification is changed.

For Lesamnta-224/256, we replace the 32 round constants described in Section 5.1.1 (page 14) of [2] with the following constants.

Round constants of [2]

0000000100000000 0000000300000002 0000000500000004 0000000700000006
0000000900000008 0000000b0000000a 0000000d0000000c 0000000f0000000e
0000001100000010 0000001300000012 0000001500000014 0000001700000016
0000001900000018 0000001b0000001a 0000001d0000001c 0000001f0000001e
0000002100000020 0000002300000022 0000002500000024 0000002700000026
0000002900000028 0000002b0000002a 0000002d0000002c 0000002f0000002e
0000003100000030 0000003300000032 0000003500000034 0000003700000036
0000003900000038 0000003b0000003a 0000003d0000003c 0000003f0000003e

New round constants

9e754700889cfedb 2db4ad503bbd6f80 02db4ad503bbd6f8 e1a70c522758bc4b
2a4989e511412ba9 1e95cf81bff8729e a8c416470af5c6d6 422bb32416c61cb6
4c85497227052110 04c8549722705211 fdf76aa9eba86421 f264994a0735e742
3744e7ab7dab9e3d 6f80451ae2875955 8b86b7ce8c169407 bda476dc1727489b
2f89be4df246d4e4 723dc79b6495eddc 966c38f97a9bdf6b 2d353aafa49d1d9b
2680aa8ac97d71b4 72ad56d717265789 1b1b82729f9e055c 90fe5ca7e52b61e3
ccd6a4153a051757 b9d177e1ac4670ae a2b05dc10bce26f5 8755b643328203fd
648150046675c089 1a79421fa88b3c2c 90e870a1365a3274 79cbdb75a8d423b5

For Lesamnta-384/512, we replace the 32 round constants described in Section 5.1.2 (page 15) of [2] with the following constants.

Round constants of [2]

0000000000000001000000000000000 0000000000000003000000000000002
0000000000000005000000000000004 0000000000000007000000000000006
0000000000000009000000000000008 000000000000000b000000000000000a
000000000000000d000000000000000c 000000000000000f000000000000000e
0000000000000011000000000000010 0000000000000013000000000000012
0000000000000015000000000000014 0000000000000017000000000000016
0000000000000019000000000000018 000000000000001b000000000000001a
000000000000001d000000000000001c 000000000000001f000000000000001e
0000000000000021000000000000020 0000000000000023000000000000022
0000000000000025000000000000024 0000000000000027000000000000026
0000000000000029000000000000028 000000000000002b000000000000002a
000000000000002d000000000000002c 000000000000002f000000000000002e
0000000000000031000000000000030 0000000000000033000000000000032
0000000000000035000000000000034 0000000000000037000000000000036
0000000000000039000000000000038 000000000000003b000000000000003a
000000000000003d000000000000003c 000000000000003f000000000000003e

New round constants

```
f6251864809494cd35cb7fa305acbe7f 78b114d45c0c003757aa6c4b9d98f1bf
b508148e2c0e460802e6cd2af27a24b0 ba220a9a4170d2de29fdd68d717f83f4
fa8e84753153428a0c9d29ba4c07bc9f 97fc92f852b9c3860d30da783d3f6b9d
95b68b70b22784abca19a58a8ca71e4c 48abbc03a30a7ff77422b58cdfd2a9ca
c7c5fa0d1976cfcbbfd178c3b7e94af7 f9c7bdd4fd083fedb7b7be15c8dcc1d3
dfc1d14920cdc088b5635cc6c7e5be34 37dcf3f822ec2133f52f774280cbc7e2
ed519add8adb45eae57e1d138887b7e1 eebfc9e5f47009f492d2f77813921014
159b340651e246363b85e6fe008b602c 2eb05b97b586d5603e4449f6e8e3f514
155b3b9423a3b0eaaf2970408e7011c9 c4acd4dbd5f51d7e0cb6c807b1a503ca
c749fd65c10030a936a9ecbe3c873d5d 58d1aa49ef6ae3f34a0cfccecddc475a
5f343b7343bca903289d46dd90e26da9 a27d71f052fa6d3232a61c086f06e116
17f09d2029b961fe360d4014031eb9db d7b2481063efc7658a41ae3d098b4854
514f4a4a1bc06c61cf87358938b8d9b4 be889af85ebc47add66113773567db05
05e3ea69155b31c85e13ac1129135b54 519d1be862b6d8976253678b149841a7
ac87ca0bc82b2705d736ec2f621c7828 2a47905563e447589bf95efede53f800
002a47905563e447589bf95efede53f8 f6e7f57d574abc562f1ea392b7ffb35b
```

The minor change of Lesamnata is only the above replcement. We will describe how to produce the new round constants in Sect. 4.2.

3 The Motivation for the Minor Change

The motivation why we propose to make a minor change to the specification of Lesamnta by changing round constants is to prevent the attacks described in [3] and one newly-discovered attack which will be described in Appendix A.1. These attacks are summarized in the following:

- Distinguishing attack on Lesamnta's block cipher [3]
  - An adversary can distinguish between Lesamnta-256's block cipher and the ideal cipher by making only two queries.
- Pseudo-collision attack on Lesamnta [3]
  - An adversary can produce a pseudo-collision of Lesamnta-256 with $O(2^{64})$ computations of the compression function.
- Semi-free start collision attack on Lesamnta [4]
  - An adversary can produce a semi-free start collision of Lesamnta-256 with $O(2^{64})$ computations of the compression function. It is a kind of pseudo-collision attack mentioned above.
- Attack using weak messages on Lesamnta
  - An adversary can find a kind of second preimage of Lesamnta-256 with $O(2^{128})$ computations of the compression function if the target message satisfies a certain property.

We observe that all the above attacks are based on some symmetry in the key scheduling function and the message mixing function of Lesamnta. To destroy the symmetry, we have chosen the new round constants. We expect that all the above attacks do not work any more for Lesamnta with the new round constants.

4 Design Principle for the New Round Constants

4.1 Condition for New Round Constants

We here show the condition allowing attacks described in Sect. 3. Since all the attacks uses it, they fail if it does not hold. We have confirmed that the new round constants do not satisfy the condition. The relationship between conditions and attacks are described in Appendix.

Let $\mathtt{C}[r][0]$ and $\mathtt{C}[r][1]$ be the left part and the right part of the r-th round constant in the key scheduling function of $EncComp_{256}$ or $EncComp_{512}$ (see Figs. 18, 28 of [2]). Note that $\mathtt{C}[r][i]$ is a 32-bit string for $EncComp_{256}$ and it is a 64-bit string for $EncComp_{512}$. We define a difference Δ_r as

$$\Delta_r = \mathtt{C}[r][0] \oplus \mathtt{C}[r][1] \tag{1}$$

for $r = 0, 1, \ldots, 31$. The condition is to satisfy all the following equations:

$$\begin{aligned}
&\Delta_0 = \Delta_4 = \Delta_8 = \Delta_{12} = \cdots = \Delta_{24} = \Delta_{28},\\
&\Delta_1 = \Delta_5 = \Delta_9 = \Delta_{13} = \cdots = \Delta_{25} = \Delta_{29},\\
&\Delta_2 = \Delta_6 = \Delta_{10} = \Delta_{14} = \cdots = \Delta_{26} = \Delta_{30},\\
&\Delta_3 = \Delta_7 = \Delta_{11} = \Delta_{15} = \cdots = \Delta_{27}.
\end{aligned} \tag{2}$$

The condition is used for distinguishing Lesamnta's block ciphers from ideal ciphers. Furthermore, the distinguishing attack can be extended to the pseudo-collision attack and the weak-message attack.

4.2 Generators of New Round Constants

To be free of suspicion of a trapdoor, round constants must be determined in a transparent way. The new round constants for Lesamnta-256 were determined by the algorihtm of Fig. 1. The algorihtm of Fig. 1 is based on the linear feedback shift register (LFSR) of the following primitive polynomial $g(x)$.

$$\begin{aligned}
g(x) = {} & x^{64} + x^{61} + x^{58} + x^{55} + x^{47} + x^{46} + x^{42} + x^{41} + x^{39} + x^{38}\\
& + x^{37} + x^{35} + x^{34} + x^{33} + x^{31} + x^{30} + x^{29} + x^{28} + x^{27} + x^{26}\\
& + x^{25} + x^{24} + x^{20} + x^{19} + x^{18} + x^{16} + x^{14} + x^{12} + x^{8} + x^{7}\\
& + x^{2} + x^{1} + 1.
\end{aligned}$$

Due to 33 non-zero coefficients, almost half of bits of the internal state may be changed by one operation. Since there are many 33-term primitive polynomials, we adopted the first 33-term primitive polynomial obtained from the decimation of the M sequence produced by the LFSR of $x^{64} + x^{63} + x^{61} + x^{60} + 1$ with the all-one initial state.

Since the round constants $\mathtt{C}[r]$ can be considered as elements of $\mathrm{GF}(2^{64})$, the above algorithm is equivalent to

$$\mathtt{C}[r] = \mathtt{C}[r-1] * \alpha^J \text{ over } \mathrm{GF}(2^{64}) \text{ for } r = 1, 2, \ldots, 31$$

```
ConstantGenerator256(word C[Nr_comp256]) /* Nr_comp256=32 */
begin
  word c = ffffffffffffffff /* in hexadecimal */
  for i = 0 to Nr_comp256*J-1  /* J = 4 */
   /* Galois-type LFSR */
   if c ∧ 0000000000000001 = 0000000000000001
     c = (c >> 1) ⊕ e18ab8ff77630124
   else
     c = c >> 1
   end if
   if i mod J = 0
     C[i/J] = c
   end if
  end for
end
```

Fig. 1. Pseudocode for generating round constants of Lesamna-224/256.

where α is a root of $g(x)$. We chose $J = 4$. If $J \leq 3$, then there exists an initial state such that almost all $\mathtt{C}[r]$'s satisfy Condition 1.

The new round constants for Lesamnta-384/512 were determined in a similar manner. Specifically, the following 65-term primitive polynomial $g(x)$ and $J = 8$ were used.

$$\begin{aligned} g(x) = {} & x^{128} + x^{124} + x^{121} + x^{120} + x^{119} + x^{117} + x^{116} + x^{114} + x^{112} \\ & + x^{111} + x^{110} + x^{107} + x^{106} + x^{105} + x^{104} + x^{101} + x^{100} + x^{98} \\ & + x^{97} + x^{95} + x^{94} + x^{93} + x^{92} + x^{91} + x^{90} + x^{89} + x^{87} + x^{86} \\ & + x^{84} + x^{82} + x^{81} + x^{79} + x^{78} + x^{76} + x^{74} + x^{73} + x^{70} + x^{69} \\ & + x^{66} + x^{64} + x^{63} + x^{60} + x^{58} + x^{54} + x^{53} + x^{51} + x^{48} + x^{39} \\ & + x^{37} + x^{36} + x^{35} + x^{32} + x^{31} + x^{30} + x^{29} + x^{28} + x^{26} + x^{23} \\ & + x^{21} + x^{18} + x^{17} + x^{15} + x^{9} + x^{8} + 1 \end{aligned}$$

The pseudocode for generating the round constants is shown in Fig. 2. Since there are many 65-term primitive polynomials, we adopted the first 65-term primitive polynomial obtained from the decimation of the M sequence produced by the LFSR of $x^{128} + x^{127} + x^{126} + x^{121} + 1$ with the all-one initial state.

5 Security against Potential Attacks on Lesamnta with New Round Constants

We here consider a hash function family D-Lesamnta which is the same as Lesamnta expept that the round constants $\mathtt{C}[r]$ are replaced by some $\mathtt{D}[r]$. We here present two potential distinguishing attacks on this Lesamnta-like hash function family. These two attacks can not work for Lesamnta with new round constants.

```
ConstantGenerator512(word C[Nr_comp512]) /* Nr_comp512=32 */
begin
  word c = ffffffffffffffffffffffffffffffff
  for i = 0 to Nr_comp512*J-1  /* J = 8 */
   /* Galois-type LFSR */
   if c ∧ 00...01 = 00...01
     c = (c >> 1) ⊕ 89dae79b7f6b6b32ca34805cfa534180
   else
     c = c >> 1
   end if
   if i mod J = 0
     C[i/J] = c
   end if
  end for
end
```

Fig. 2. Pseudocode for generating round constants of Lesamna-384/512.

Condition 1 Let $a = a_0 \| a_1 \| \ldots \| a_7$ and $b = b_0 \| b_1 \| \ldots \| b_7$, where $a_i, b_i \in \{0,1\}^8$. Let $\mathsf{tp}_\ell(a) = b$ be a byte-transposition such that $b_i = a_{i+2\ell \bmod 8}$ for $0 \le \ell \le 3$. Let $\mathsf{rv}(a) = b$ be a byte-transposition such that $b_i = a_{7-i}$.

We have a distinguishing attack on the underlying block cipher of the D-Lesamnta-256 output function if D satisfies the following condition.

Let $\mathtt{D}[r]$ be the 64-bit r-th round constant in the key scheduling function of $EncOut_{256}$. $\mathtt{D}[r]$ is considered as an 8-byte data, that is,

$$\mathtt{D}[r] = \mathtt{D}[r](0) \parallel \mathtt{D}[r](1) \parallel \ldots \parallel \mathtt{D}[r](7),$$

where $\mathtt{D}[r](i) \in \{0,1\}^8$. We define a 64-bit difference Λ_r as

$$\Lambda_r = \mathtt{D}[r] \oplus \Pi(\mathtt{D}[r])$$

for $r = 0, 1, \ldots, 31$, where Π is any composition of tp_ℓ and rv but $\Pi \neq \mathsf{tp}_0$.

The condition is to satisfy the following equations:

$$\Lambda_k = \Lambda_{k+4} \quad \text{for } 0 \le k \le 26 \ .$$

Notice that round constants of $EncOut_{256}$ are identical to those of $EncComp_{256}$ (see Fig. 20 of [2]).

Since the new round constants for Lesamnta-256 do not satisfy the above condition, Lesamnta-256 is secure against the above attack.

Condition 2 Let $s = (s_{i,j})$ and $t = (t_{i,j})$, where $s_{i,j}, t_{i,j} \in \{0,1\}^8$ and $0 \le i, j \le 3$. Let $\pi_\ell(s) = t$ be a byte-transposition such that $t_{i,j} = s_{i+\ell \bmod 4,j}$ for $0 \le \ell \le 3$. Let $\varpi_\ell(s) = t$ be a byte-transposition such that $t_{i,j} = s_{i,j+\ell \bmod 4}$ for $0 \le \ell \le 3$.

We have a distinguishing attack on the underlying block cipher of the D-Lesamnta-512 output function if D satisfies the following condition.

Let $\mathtt{D}[r]$ be the 128-bit r-th round constant in the key scheduling function of $EncOut_{512}$. $\mathtt{D}[r]$ is considered as a 16-byte data, that is,

$$\mathtt{D}[r] = \mathtt{D}[r](0) \parallel \mathtt{D}[r](1) \parallel \ldots \parallel \mathtt{D}[r](15),$$

where $\mathtt{D}[r](i) \in \{0,1\}^8$. We see $\mathtt{D}[r] = (\mathtt{D}[r]_{i,j})$, where $\mathtt{D}[r]_{i,j} = \mathtt{D}[r](i+4j)$ for $0 \le i, j \le 3$. We define a 128-bit difference Ξ_r as

$$\Xi_r = \mathtt{D}[r] \oplus \Pi(\mathtt{D}[r])$$

for $r = 0, 1, \ldots, 31$, where Π is any composition of π_ℓ and $\varpi_{\ell'}$ but Π is not an identity transposition. The condition is to satisfy the following equations:

$$\Xi_k = \Xi_{k+4} \quad \text{for } 0 \le k \le 26 \ .$$

Notice that round constants of $EncOut_{512}$ are identical to those of $EncComp_{512}$ (see Fig. 28 of [2]).

Since the new round constants for Lesamnta-512 do not satisfy the above condition, Lesamnta-512 is secure against the above attack.

6 The Impact of the Change on Lesamnta

6.1 Impact on the Security of Lesamnta

We observe that the change does not cause any impact on resistance against the known attacks on Lesamnta described in [2].

We expect that Lesamnta with the new round constants prevents the attacks described in Sect. 3 which violated the assumptions of the security proofs in the ideal cipher model in [2]. Therefore we believe that these security proofs for Lesamnta would become more meaningful if the round constants are changed to the new ones.

6.2 Impact on the Performance of Lesamnta

For speed-optimized implementations of Lesamnta, the change does not cause any impact on its performance because the round constants are stored in a table.

For area-optimized implementations of Lesamnta, the change may slightly increase the required memory size because on-the-fly generation of round constants may be less useful than storing them in a table, due to the relatively large number of terms in the feedback polynomial used in the LFSR generating them. However, based on our estimation, we expect that storing them in a table does not cause any problem in real applications.

7 Concluding Remarks

We propose to make a minor change to the specification of Lesamnta by changing round constants. The motivation of this change is to prevent the attacks described in [3] and the newly-discovered attack described in this document.

We expect that all of these attacks do not work any more for Lesamnta with the new round constants. We also expect that the change does not cause any significant impact on resistance against the known attacks and on performance of Lesamnta.

Acknowledgments

We would like to thank Charles Bouillaguet, Orr Dunkelman, Gaëtan Leurent, Pierre-Alain Fouque for their excellent analysis on Lesamnta. We would like to thank Mridul Nandi for improving the analysis. We would like to mention the people who gave us feedback and important comments on this work: Kota Ideguchi, Yasuko Fukuzawa, Toru Owada, Bart Preneel. This work was partially supported by the National Institute of Information and Communications Technology, Japan.

References

1. C. Bouillaguet, O. Dunkelman, G. Leurent, and P. A. Fouque, Private communication, 2009.
2. S. Hirose, H. Kuwakado, and H. Yoshida, "SHA-3 proposal: Lesamnta," `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/Lesamnta.zip%`, October 2008. latest version: `http://www.sdl.hitachi.co.jp/crypto/lesamnta/`.
3. S. Hirose, H. Kuwakado, and H. Yoshida, "Security Analysis of the Compression Function of Lesamnta and its Impact," `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/LESAMNTA_Comments.pdf`, June 2009.
4. M. Nandi, Private communication, 2009.
5. National Institute of Standards and Technology, "Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family," `http://csrc.nist.gov/groups/ST/hash/documents/`, November 2007.

8 List of Annexes

A The Attack Methods

A.1 Attack Using Weak Messages on Lesamnta

The pseudo-collision-finding attack might be applicable to an attack using weak messages. The primary idea of this attack was shown in [1].

Consider Lesamnta-256. The algorithm of an adversary that finds a second preimage is described below.

1. Suppose that a t block message and its digest are given. The t-block message is denoted by $\mathtt{mb}^{(0)} \parallel \mathtt{mb}^{(1)} \parallel \ldots \parallel \mathtt{mb}^{(t-1)}$ where $\mathtt{mb}^{(i)}$ is a 256-bit message block.
2. For $i = 0, 1, \ldots, t-1$, compute $\mathtt{chain}^{(i)}$ as
$$\mathtt{chain}^{(i)} = \mathtt{Compression256}(\mathtt{chain}^{(i-1)}, \mathtt{mb}^{(i)}),$$
where $\mathtt{chain}^{(-1)}$ is the standard initial value.
3. If there is an index i_1 such that
$$\mathtt{chain}^{(i_1)}[j] = \mathtt{chain}^{(i_1)}[j+1]$$
for $j = 0, 2, 4, 6$, go to the next step. Otherwise output fail.
4. Find $\mathtt{mb}'^{(i_1-1)}$ and $\mathtt{mb}'^{(i_1)}$ such that
$$\begin{aligned} \mathtt{chain}'^{(i_1-1)} &= \mathtt{Compression256}(\mathtt{chain}^{(i_1-2)}, \mathtt{mb}'^{(i_1-1)}), \\ \mathtt{chain}^{(i_1)} &= \mathtt{Compression256}(\mathtt{chain}'^{(i_1-1)}, \mathtt{mb}'^{(i_1)}), \end{aligned}$$
by using the attack described in [3].

The probability that the condition in step 3 is satisfied is $t/2^{128}$. Since Lesamnta-256 accepts a $(2^{64}-1)$-bit message at most, the probability is less than 2^{-72}. We call a message satisfying the condition in step 3 a *weak message*. We notice that this attack is effective only when a given message is a weak message.