From: nasoor bagheri <na.bagheri@gmail.com>
Sent: Monday, May 06, 2019 3:20 AM
To: lwc-forum@list.nist.gov
Cc: Danilo Gligoroski; sadegh sadeghi; Majid Mahmoudzadeh Niknam
Subject: [lwc-forum] Official comment:GAGE AEAD

Dear All,

In GAGE, in the aead mode, we have |T|=128. On the other hand, for some variants, i.e. b=234 and b=240, b- |T|< |T|. In such case, given T, the adversary can just guess the remaining part to recover the state and so do a forgery attack, for example. Hence, we believe the claimed security in table 2.1, for these variants are not correct.

Best Regards,
Nasour Bagheri, Sadegh Sadeghi and Majid Niknam

--

From: Danilo Gligoroski <danilog@ntnu.no>
Sent: Monday, May 06, 2019 5:58 PM
To: nasoor bagheri; lwc-forum@list.nist.gov
Cc: sadegh sadeghi; Majid Mahmoudzadeh Niknam
Subject: [lwc-forum] Re: Official comment:GAGE AEAD

Dear Nasour, Sadegh and Majid,

Thank you for your continuous interest and analysis of GAGE and InGAGE.

If I understand correctly your remark, your forgery attack by guessing the remaining part of the state assumes a "Nonce reuse", right?

Regards,

Danilo!

On 06/05/2019 03:19, nasoor bagheri wrote:

> Dear All,
>
> In GAGE, in the aead mode, we have |T|=128. On the other hand, for some variants, i.e. b=234 and b=240, b- |T|< |T|. In such case, given T, the adversary can just guess the remaining part to recover the state and so do a forgery attack, for example. Hence, we believe the claimed security in table 2.1, for these variants are not correct.
>
> Best Regards,
> Nasour Bagheri, Sadegh Sadeghi and Majid Niknam

From: nasoor bagheri <na.bagheri@gmail.com>
Sent: Tuesday, May 07, 2019 2:46 AM
To: Danilo Gligoroski
Cc: lwc-forum@list.nist.gov; sadegh sadeghi; Majid Mahmoudzadeh Niknam
Subject: [lwc-forum] Re: Official comment:GAGE AEAD

Dear Danilo,

Thank you for your reply. The presented remark works even in nonce respecting setting. e.g. consider the below scenario:
1) given (A, T, N), assuming |P|=0, i.e. empty plaintext, and |A|> b- |T|.
2) then the adversary guesses the missing b- |T| bits of the last permutation to retrieve the state, where it is possible to use the associated data A to filter wrong guesses.
3) Given the state, then it is easy to generate the valid (A, P, C, T, N) for any desired P.
the complexity would be 2^{b- |T|} which is less than 2^{|T|} when |T|=128 and b=232 or b= 240.
Please note that the user has not repeated the nonce and henceforth the above scenario does not violate the nonce respecting assumption.
To us, to fix this point, either the key should have been used in the last block, similar to some other schemes, or the security claim should be reduced for those variants.

Please correct us if are missing any point.

Best Regards,
Nasour, Sadegh and Majid

On Tue, May 7, 2019 at 2:27 AM Danilo Gligoroski <danilog@ntnu.no> wrote:

> Dear Nasour, Sadegh and Majid,
>
> Thank you for your continuous interest and analysis of GAGE and InGAGE.
>
> If I understand correctly your remark, your forgery attack by guessing the remaining part of the state assumes a "Nonce reuse", right?
>
> Regards,
>
> Danilo!
>
> On 06/05/2019 03:19, nasoor bagheri wrote:
>
> > Dear All,
> >
> > In GAGE, in the aead mode, we have |T|=128. On the other hand, for some variants, i.e. b=234 and

From: Danilo Gligoroski <danilog@ntnu.no>
Sent: Tuesday, May 07, 2019 9:39 AM
To: lwc-forum@list.nist.gov
Subject: Re: [lwc-forum] Re: Official comment:GAGE AEAD

Dear Nasour, Sadegh and Majid,

Yes, we will update the Table 2.1 for b=232 and b=240.

Thank you very much for your valuable input,

Danilo!

On 07/05/2019 02:46, nasoor bagheri wrote:

> Dear Danilo,
>
> Thank you for your reply. The presented remark works even in nonce respecting setting. e.g. consider the below scenario:
> 1) given (A, T, N), assuming |P|=0, i.e. empty plaintext, and |A|> b- |T|.
> 2) then the adversary guesses the missing b- |T| bits of the last permutation to retrieve the state, where it is possible to use the associated data A to filter wrong guesses.
> 3) Given the state, then it is easy to generate the valid (A, P, C, T, N) for any desired P.
> the complexity would be 2^{b- |T|} which is less than 2^{|T|} when |T|=128 and b=232 or b= 240.
> Please note that the user has not repeated the nonce and henceforth the above scenario does not violate the nonce respecting assumption.
> To us, to fix this point, either the key should have been used in the last block, similar to some other schemes, or the security claim should be reduced for those variants.
>
> Please correct us if are missing any point.
>
> Best Regards,
> Nasour, Sadegh and Majid
>
> On Tue, May 7, 2019 at 2:27 AM Danilo Gligoroski <danilog@ntnu.no> wrote:
>
> > Dear Nasour, Sadegh and Majid,
> >
> > Thank you for your continuous interest and analysis of GAGE and InGAGE.

From: nasoor bagheri <na.bagheri@gmail.com>
Sent: Tuesday, May 07, 2019 10:30 AM
To: Danilo Gligoroski
Cc: lwc-forum@list.nist.gov; sadegh sadeghi; Majid Mahmoudzadeh Niknam
Subject: Re: [lwc-forum] Re: Official comment:GAGE AEAD

Dear Danilo,

Thank you for the feedback.

Best Regards,
Nasour, sadegh and Majid

On Tue, May 7, 2019, 6:09 PM Danilo Gligoroski <danilog@ntnu.no> wrote:

> Dear Nasour, Sadegh and Majid,
>
> Yes, we will update the Table 2.1 for b=232 and b=240.
>
> Thank you very much for your valuable input,
>
> Danilo!
>
> On 07/05/2019 02:46, nasoor bagheri wrote:
>
> > Dear Danilo,
> >
> > Thank you for your reply. The presented remark works even in nonce respecting setting. e.g. consider the below scenario:
> > 1) given (A, T, N), assuming |P|=0, i.e. empty plaintext, and |A|> b- |T|.
> > 2) then the adversary guesses the missing b- |T| bits of the last permutation to retrieve the state, where it is possible to use the associated data A to filter wrong guesses.
> > 3) Given the state, then it is easy to generate the valid (A, P, C, T, N) for any desired P.
> > the complexity would be 2^{b- |T|} which is less than 2^{|T|} when |T|=128 and b=232 or b= 240.
> > Please note that the user has not repeated the nonce and henceforth the above scenario does not violate the nonce respecting assumption.
> > To us, to fix this point, either the key should have been used in the last block, similar to some other schemes, or the security claim should be reduced for those variants.
> >
> > Please correct us if are missing any point.
> >
> > Best Regards,
> > Nasour, Sadegh and Majid

From: Danilo Gligoroski <danilog@ntnu.no>
Sent: Tuesday, May 07, 2019 7:48 PM
To: lightweight-crypto
Cc: lwc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: GAGE and InGAGE

Dear all,

We have updated Table 1.4 and Table 2.1 in the document for GAGE and InGAGE and made some redacting changes.

The updated document can be taken from the newly register web page http://gageingage.org/ i. e. from http://gageingage.org/upload/GAGEandInGAGEv1.01.pdf

Algorithm specifications have not been changed.

Change log is also included in the document.

We thank Nasour Bagheri, Sadegh Sadeghi and Majid Niknam for their valuable input.

Best regards,

GAGE and InGAGE team

P.S. NIST people can now add a link for our website: http://gageingage.org/

From: Danilo Gligoroski <danilog@ntnu.no>
Sent: Thursday, August 1, 2019 3:24 PM
To: lightweight-crypto
Cc: lwc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: GAGE and InGAGE

Hi,

We would like to report on our first FPGA implementation of the smallest member of GAGE family of hash functions: GAGE256 with b=232, c=224 and r=8.

The synthesis on Artix-7 device: xc7a35tcpg236-1 on all-parallel implementation use 402 LUTs, 516 Flip Flops and 0 RAM resources, running at 250 MHz.

The VHDL code can be taken from the web page: http://gageingage.org/

We would like to inform that the latest document for GAGE and InGAGE is v1.03 and can be taken from: http://gageingage.org/upload/GAGEandInGAGEv1.03.pdf

Several typos have been corrected (no change in source code or test vectors), a new section about hardware implementations has been added, and Mohamed El-Hadedy has been added as a new member of GAGE team.

Best regards,

Danilo!

From: Danilo Gligoroski <danilog@ntnu.no>
Sent: Thursday, August 22, 2019 10:37 AM
To: lightweight-crypto
Cc: lwc-forum@list.nist.gov
Subject: OFFICIAL COMMENT: GAGE and InGAGE

Hi,

I would like to report that Mohamed El-Hadedy have produced a new sequential FPGA implementation for GAGE256 with b=232, c=224 and r=8.

The synthesis on Artix-7 device: xc7a35tcpg236-1 on all-sequential implementation use 226 LUTs, 120 Flip Flops and 0 RAM resources.

The VHDL code can be taken from the web page: http://gageingage.org/

Best regards,

Danilo!

From: bg <bg@nerilex.org>
Sent: Thursday, August 22, 2019 2:42 PM
To: lightweight-crypto; lwc-forum@list.nist.gov
Subject: [lwc-forum] OFFICIAL COMMENT: GAGE and InGAGE
Attachments: signature.asc

Hi,

we recently published new optimized implementations of the cryptographic sponge function in GAGE for AVR microcontrollers.

You can download the code on our web page:
http://gageingage.org

These following table gives a rough idea of the compactness of the core function.

Name	Cycles	RAM	ROM
gage-core-232 A	215273	33	218
gage-core-232 B	34642	8	478
gage-core-256 A	237437	36	218
gage-core-256 B	32778	8	466

The B variant is a bitsliced implementation which is significantly faster while still being compact.

Our work on those implementations is still ongoing and further improvements will be published soon.

But we want to offer figures which offer a rough idea of what is possible with GAGE and InGAGE now.

Best regards,
~ bg

--
bg nerilex
Daniel Otte
E-Mail: bg@nerilex.org
XMPP/Jabber: bg@nerilex.org
Mastodon: https://bg@naos.crypto.church
Fingerprint: CB4E 915F ACAD EEC2 0D34 D266 2978 788D 0DB2 E18E