A New Scheme for Authenticated Encryption

STSM Project report, July 2013

ICT COST action IC1204

Trustworthy Manufacturing and Utilization of Secure Devices

Hristina Mihajloska

Host Institution: Norwegian University of Science and Technology, Department of Telematics, Prof. Danilo Gligoroski, danilog@item.ntnu.no

Visitor: Hristina Mihajloska, PhD student at Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, hristina.mihajloska@finki.ukim.mk

Visiting Period: from June 17, 2013 to July 19, 2013 (1 month).

Purpose of the STSM:

Authenticated encryption has long been a vital operation in cryptography by its ability to provide confidentiality, integrity and authenticity at the same time. Most of the existing schemes have been designed with software platforms in mind; therefore they are not always hardware friendly. This question has led to the recently-announced authenticated encryption competition CAESAR.

CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) has terms Applicability and Robustness which means that the new cipher has to have widespread adoption, to be also suitable for lightweight designs, and to be prone on insider attacks when the key is known. This is not a case with the existing standard AES-GCM, so that's why the main target of this competition is to come up with an authenticated encryption scheme that offers these advantages over it. The design of an authenticated encryption scheme is a trade-off between many different aspects such as security, speed, area, and power. Existing schemes for authenticated encryption can hardly fulfill more than one or two of these criteria at the same time. Furthermore, most of them are derived from block cipher modes of operation and designed to work efficiently with only AES. In this project, we aim to bridge this gap by providing a new authenticated encryption scheme that is not only hardware efficient, but also scalable.

At the Department of Telematics of NTNU there is a strong research group in cryptography and information security. From their recent activities it is notable their participation in SHA-3 hash competition with two functions (Edon-R and BMW) where Blue Midnight Wish (BMW) hashing algorithm designed by Prof. Gligoroski and his team entered the second round of the SHA-3 competition.

The research group of FCSE has done intense research in the field of symmetric cryptography in collaboration with the researchers of NTNU in the last 10 years. One of the main results of this research is the development of eSTREAM finalist Edon-R.

The planned research of on a new authenticated cipher is a perfect problem within the context of NTNU's research resources, while it may also lead to new models for effective representations of AE schemes in hardware. It will also set a perfect example to international collaboration between researchers from EU and non EU countries.

Description of the work:

During my STSM at NTNU, Department of Telematics, Trondheim, Norway, several aspects of designing authenticated encryption with associated data (AEAD) ciphers have been addressed; the research can be grouped in the following activities:

- writing a white paper for DIAC 2013: Directions in Authenticated Ciphers, which will be held on 11-13 August 2013, Chicago, USA (attached as Appendix 1)
    - D. Gligoroski, H. Mihajloska and H. Jakobsen: *Should MAC's retain hash properties when the key is known in the next AEAD?*
- writing a panel proposal for the new idea of AEAD design for DIAC 2013: Directions in Authenticated Ciphers, which will be held on 11-13 August 2013, Chicago, USA (attached as Appendix 2)
    - D. Gligoroski, H. Mihajloska and H. Jakobsen: *OWCM: One-Way Counter Mode (initial design)*.
- developing a lightweight version of our AEAD scheme.

After having discussion with the main organizer of DIAC 2013, Dan Bernstein, both of our proposals will be presented in one presentation slot under the title: OWCM: One-Way Counter Mode.

With the first one, our goal is to initiate a discussion at DIAC 2013 about the AEAD ciphers with the following property: "*Users of the cipher can easily find different messages that produce same authentication tags*". We offer several realistic scenarios how to exploit this property in an AEAD cipher. As an easy exercise we describe how one of the communicating parties that posses the secret key can find different messages that give same authentication tag for CCM, GCM and OCB. We point out that none of these scenarios can happen if the authentication is done by the use of cryptographic hash functions such as new SHA-3 or the older HMAC scheme. This final point raises again the necessity of having ultra-fast one-way cryptographic functions.

In the second one, we present the initial design of our AEAD scheme based on the counter mode combined with a use of one-way compression functions. Our proposal can be seen as a modification of the GCM scheme, where the operations in GHASH are replaced by a use of a double-pipe one-way compression function. The way how we combine the use of the one-way compression function is similar as that used in the HMAC scheme. Our goal was to design a robust AEAD that will offer the uniqueness of the MAC tags even if the secret key is revealed and to resemble the design style of HMAC that was confirmed as a secure MAC even when the used hash function is not a collision resistant (as long as it is a preimage resistant). The specific construction of the used one-way

function and its efficiency is still under our investigation. First, we would like to hear comments, critique and suggestions from fellow cryptographers attending DIAC 2013.

The third activity is still on-going and is expected that at the end of the summer we will have the complete implementation of our new lightweight AEAD design. Our design is based on a recently most popular SHA-3 winner, Keccak family of cryptographic sponge functions.

Sponge construction or sponge function is a class of algorithms with finite state that take an input bit stream of any length and produce an output of any desired length. It can be used to model or implement most cryptographic primitives, including cryptographic hash functions, message authentication codes, stream ciphers, block ciphers and pseudo number-random generators. This function is built from three components: state memory, permutation function and padding function (Figure 1).

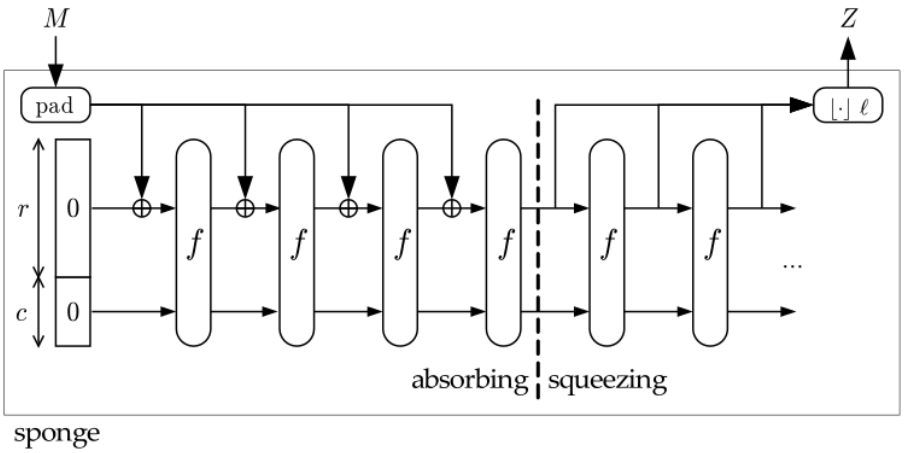


Figure 1. The sponge construction

The most interesting and core part is the permutation component *f*, because the whole security of the primitive relay on it. Designing cryptographically strong permutation suitable for such purposes is similar to designing a block cipher without the key scheduling part.

In our new lightweight AEAD design we decided to work on construction of good permutation function, which will be not just cryptographically strong, but also suitable for lightweight designs. In the symmetric cryptography, two design paradigms of permutation functions are known, SP-network and ARX design. The ARX design philosophy is opposed to SP-network design. Instead of using S-boxes for increasing the non-linearity of the input bits and permutation layer for their diffusion, ARX design only uses Additions, Rotations and Xors. These operations are very simple on the bit level and can be relatively fast and cheap implemented in hardware and software, they run in constant time and are prone to timing attacks. Also when you mixed together on an appropriate way, they interact in a complex and non-linear way. However most of the known symmetric-key cryptographic primitives are based on the SP-network design paradigm, because it is well known and good understood, but ARX design is much more promising and efficient for implementation in hardware and software. The later can be supported that two of the five finalists in the recently SHA-3 competition were based on ARX constructions.

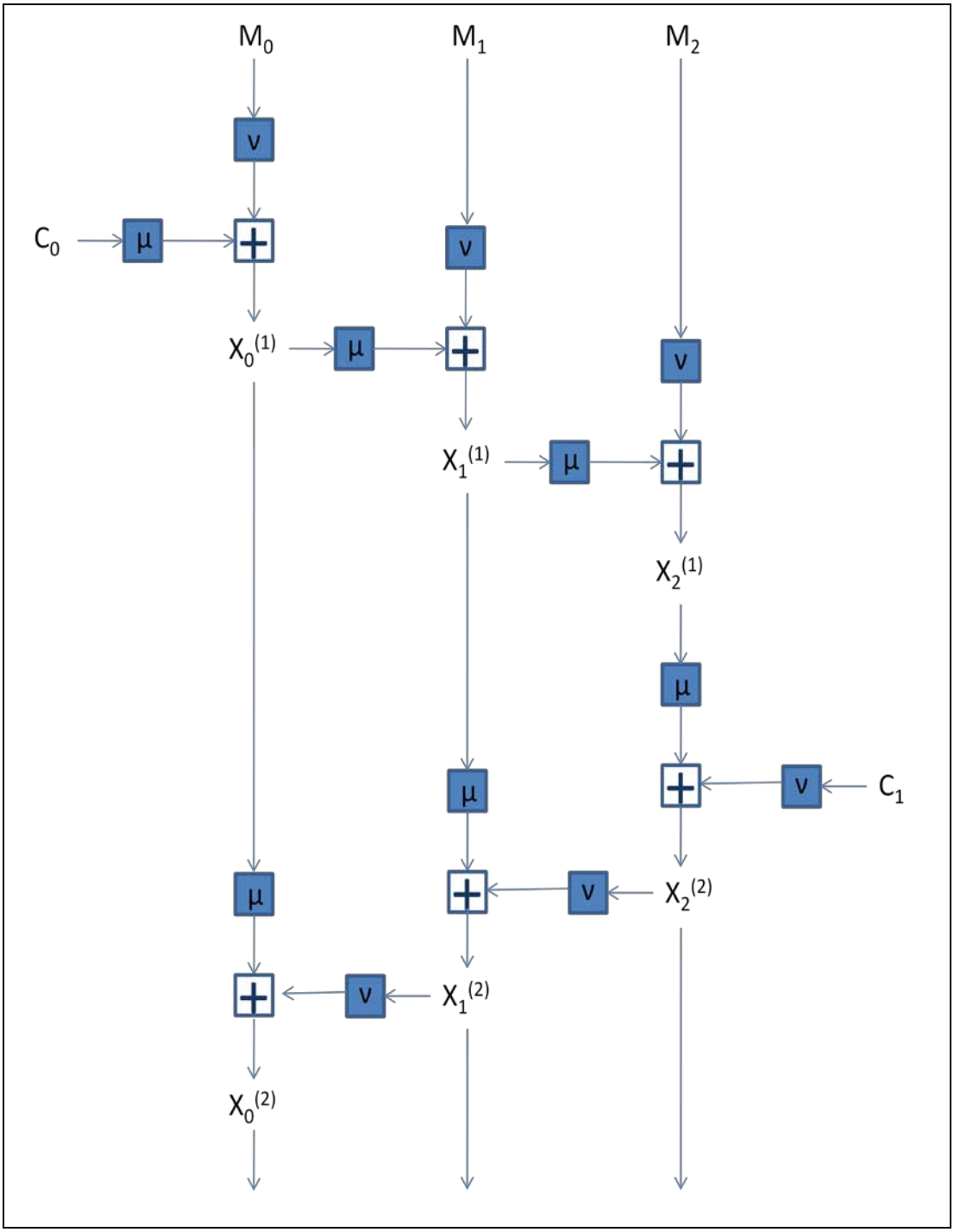


Figure 2. One round of permutation function

According to the previous research work of Prof. Gligoroski and his contribution and experience at the SHA-3 competition, the ARX design paradigm for the permutation function has been seen very suitable for our new design. In this work we focused on the construction of lightweight version of authenticated encryption cipher based on sponge function with ARX design.

The initial design is done; one round of it can be seen on the Figure 2. This design is 192-bit; each message block has 192 bits, which are represented as a sequence of twelve 16-bit words. The

underlying function is a permutation, denoted as $f[b]$, where $b = 192$ is the width of the permutation. The width of the permutation is also the width of the state in the sponge construction. The state is organized as a list of three 4-tuples, each of length $w = 16$, ($b = 12w$). We obtain our sponge function design with parameters, capacity $c = 160$ and bitrate $r = 32$ bits in order to achieve minimum provable security of preimage attack of 2^{80}. The number of rounds n_r we decided to put as a tweakable parameter, so in the first design it will be 8.

The general permutation is consisted of two main transformations μ and ν. The following operations are applied:

1. Bitwise logic word operations $\oplus$ - XOR;
2. Addition + modulo 2^{16};
3. Rotate left (circular left shift) operation, $ROTL^r(x)$, where x is a 16-bit word and r is an integer with $0 \leq r < 16$.

Transformations μ and ν are Boolean maps from $\mathbb{F}_2^{64} \rightarrow \mathbb{F}_2^{64}$, where $\mathbb{F}_2$ is a Galois field with two elements. They do the work of diffusion and nonlinear mixing of the input bits.

Our goal is to make the design provably resistant against linear and differential cryptanalysis.

Future collaboration:

During my STSM at NTNU, our research work gave birth to positive results which encouraged and motivated us to continue with further research in the field of lightweight authenticated encryption ciphers and with it, general in the field of symmetric cryptography. The next step will be improving and optimizing the hardware implementation of our design on 16-bit registers. Also we are planning to adapt our design for 8-bit registers to be more suitable for embedded devices. We presume that our hardware implementation will be more compact, securely and will require less GEs then existing ones, which is the main point of the lightweight cryptography.
The continued research would focus on creating a family of AEAD ciphers (lightweight and regular) that will be able to stand back to back with the world's best algorithms for AEAD designs.

Appendix 1

Should MAC's retain hash properties when the key is known in the next AEAD?

Danilo Gligoroski[1] and Hristina Mihajloska[2] and Håkon Jacobsen[1]

[1] Department of Telematics, Norwegian University of Science and Technology (NTNU), Trondheim, NORWAY, {danilog, hakoja}@item.ntnu.no
[2] "Ss Cyril and Methodius" University, Faculty of Computer Science and Engineering (FINKI), Skopje, MACEDONIA, hristina.mihajloska@finki.ukim.mk

Abstract. The purpose of this note is to initiate a discussion at DIAC 2013 about the AEAD ciphers with the following property: "*Users of the cipher can easily find different messages that produce same authentication tags.*". We offer several realistic scenarios how to exploit this property in an AEAD cipher. As an easy exercise we describe how one of the communicating parties that posses the secret key can find different messages that give same authentication tag for CCM, GCM and OCB. We point out that none of these scenarios can happen if the authentication is done by the use of cryptographic hash functions such as new SHA-3 or the older HMAC scheme. This final point raise again the necessity of having ultra-fast one-way cryptographic functions.

1 Introduction

Cryptographic literature (for example *Handbook of Applied Cryptography* [10]) dealing with the problems of authenticated encryption considers schemes that provide:

- Message authentication that provide data origin authentication with respect to the original message source (and data integrity, *but no uniqueness*).
- Message authentication that provide data origin authentication with respect to the original message source (and data integrity, *AND uniqueness*) - in [10, Remark 9.8, pp. 325] referred to as *MAC resistance with known key.*

Most existing security models for AE [3] and its newer variant AEAD [2] have security proofs where the forgery attempts are done by third parties i.e., the models are designed to detect intentional, unauthorized modifications of the data, and accidental modifications but only by third parties.

However, as noticed in [10, Remark 9.8, pp. 325] if the authentication is performed with a cryptographic hash function[3], for example in the standard *Encrypt-then-MAC* (EtM) scheme using HMAC [8], then the scope of protection against intentional and unauthorized modifications can be meaningfully

[3] There referred to with the abbreviation MDC - Modification Detection Codes.

increased to also include the communicating parties (which holds the keys). In particular, we might want the MAC to retain some of the properties of a hash function when the key is known. We will give examples of some scenarios where this feature can be useful in Section 2.

Unfortunately, the property of degrading to a hash function under a known key (as in EtM-with-HMAC), can lead to a significant drop in efficiency. To address the need for an efficient AEAD scheme, several schemes have been proposed (OCB [7]) and standardized (CCM [15] and GCM [9]). In these models the problem of non-trusting communicating parties is not addressed at all. Note, that this makes the implicit assumption that the communicating parties trust each other, or the mutual integrity of the messages have to be guaranteed by other cryptographic mechanisms such as digital signatures or HMACs, which return us to the first situation of using the slower EtM-with-HMAC.

CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) [5] has the term *Robustness* as one of its main goals. Our position is that authenticated ciphers that do not offer distinct tags for distinct messages are possibly not robust ciphers. Additionally, when AEAD schemes are used in protocols and applications, implementers might wrongly assume that different messages will always lead to different tags.

We would like to initiate a discussion on whether this is an issue that should be considered for CEASAR submissions. To paraphrase Bernstein [6] from his recent post to the `crypto-competitions@googlegroups.com` mailing-list (discussing some other requirements, but the statement is also applicable for the issues we discuss in this note):

> "Ignoring these requirements doesn't make them go away. Ciphers that fail to solve the problems simply force users to deploy their own solutions, producing a complicated, fragile system, whereas it's relatively easy to integrate solutions directly into the ciphers."

2 Exploits of AEAD ciphers with easy tag collisions

2.1 Exploit in "Secure audit logs"

The following exploit scenario is adopted from the Bellare-Yee article [4] about the "Secure audit logs". An attacker is breaking into a machine that keeps activity logs that are encrypted by an AEAD scheme. He/she has obtained the encryption key by some other means (physical force, stealing, ...). In order to protect against such accidental revelation of encryption keys, the authentication tags are kept in a separate and write protected area. This way the existing encrypted logs are protected from being overwritten with other fake logs. However, if the AEAD scheme was implemented by CCM, GCM or OCB, the attacker can erase his/her previous (unsuccessful) attempts to break-in by simply producing a log file that has the same authentication tag as the originally encrypted log.

2.2 Multi-cast authentication

The following scenario is an adoption of the *multidestination secure mail problem*, discussed by Mitchell and Walker [12, 11], to the setting of ciphers providing authenticated encryption.

Suppose Alice wants to send an authenticated message to Bob and Claire in a group chat application. Assume further that all group communication goes through a central hub which relays a single message from one party to the other two. At the start of the session the application establishes pairwise symmetric keys among the participants, i.e. Alice and Bob shares the key K_{AB}, Alice and Claire shares the key K_{AC} and Bob and Claire shares K_{BC}. Exactly how these keys are established is immaterial.

Assume that the chat application employs an AEAD scheme. Following the notation of [13], an AEAD scheme is a function $\mathcal{E}\colon \mathcal{K}\times\mathcal{T}\times\mathcal{A}\times\mathcal{S}\times\mathcal{M}\to\{0,1\}^*$, over the space of *keys*, *public message numbers*, *associated data*, *secret message numbers* and *messages*. We write this function more compactly as:

$$C \parallel T \leftarrow \mathcal{E}_K^{N,A}(S, M),$$

where $C \parallel T$ denotes that the ciphertext can be parsed into an "encryption-part" and a "tag-part". Since the nonce and associated data are not very relevant for this exposition, we will for clarity ignore them in the above notation and simply write $\mathcal{E}_K(M)$.

In order to send the message X to both Bob and Claire, Alice proceeds as follows:

1. She selects a session key $K_S \in \mathcal{K}$ which will be used for this one message only.
2. She computes the ciphertext $C \parallel T \leftarrow \mathcal{E}_{K_S}(X)$.
3. Using the keys she shares with Bob and Claire individually, Alice prepares the following message which is sent to both Bob and Claire:

$$C \parallel \mathcal{E}_{K_{AB}}(K_S \parallel T) \parallel \mathcal{E}_{K_{AC}}(K_S \parallel T). \tag{1}$$

When Bob and Claire receive this message they can extract K_S and T using the keys they share with Alice, and verify the authenticity of the message.

The idea of this scheme is that since $\mathcal{E}$ is an authenticated encryption scheme, Claire cannot modify C without it being detected by T. Additionally, she cannot simply recompute a new ciphertext with the key K_S, since she cannot get to the old T value encrypted with the key shared between Alice and Bob.

Unfortunately, if the authenticated encryption cipher makes it easy to find colliding tags for different messages when the key is known, the above scheme can easily be broken. In particular, Claire can spoof a message to Bob as if it came from Alice. For instance, if the application uses any one of CCM, GCM or OCB as its authenticated cipher, Claire can create another ciphertext $C' \parallel T$, with $C' \neq C$, using any of the techniques described in Section 3. Assuming Claire is able to intercept the message from the hub to Bob, she can swap C with C' in (1) and Bob will accept this to be a valid message from Alice.

3 Examples: How to find tag collisions for CCM, GCM and OCB

3.1 How to find tag collisions for CCM

CCM is the abbreviation for Counter with Cipher Block Chaining-Message Authentication Code [15]. CCM authenticated encryption with associated data basically combines the counter (CTR) mode for a data encryption and CBC-MAC mode for a data authentication. It works only with an approved symmetric key block cipher algorithm whose block size is 128 bits, such as AES [1]. Only one underlying key is used for both the authentication and the encryption part. The input to CCM includes three elements: 1) data that will be both authenticated and encrypted, called the payload P; 2) associated data, A that will be authenticated but not encrypted; and 3) a unique value, called a nonce N, that is assigned to the payload and the associated data.

To process each message block, a counter is encrypted with the underlying block cipher and the result is XORed to the message for ciphertext production. The message is also XORed with the accumulator which is then encrypted. The accumulated value corresponds to the internal message authentication state, and is kept being accumulated and updated until all the messages are processed. After all blocks have been processed, the output is XORed with the first encrypted nonce, producing the authentication tag. At the end of processing of each message block, the counter is also incremented for the next message block encryption.

We give one scenario how to find colliding messages for CCM.

If we choose two consecutive message blocks P_1 and P_2, the formulas for the internal message authentication state are given below:

$$X_1 = \mathcal{E}_K(X_0 \oplus P_1)$$
$$X_2 = \mathcal{E}_K(X_1 \oplus P_2)$$

where X_0 is the accumulated value from the previous internal message authentication state. If we replace the first message block P_1 with an arbitrary new value P_1' then we have:

$$X_1' = \mathcal{E}_K(X_0 \oplus P_1')$$

In the next state we want to produce the same authenticated value X_2, but now with the new value X_1' and a second message block P_2' that will compensate the introduced new value of P_1':

$$X_2 = \mathcal{E}_K(X_1' \oplus P_2').$$

Thus,

$$X_1 \oplus P_2 = X_1' \oplus P_2',$$

so, P_2' should be:

$$P_2' = X_1 \oplus P_2 \oplus \mathcal{E}_K(X_0 \oplus P_1'). \tag{2}$$

3.2 How to find tag collisions for GCM

GCM is the abbreviation for Galois/Counter Mode [9]. The encryption stage is similar to CCM, but authentication is realized via universal hashing of the produced ciphertext blocks over a binary Galois Field $GF(2^{128})$ instead of the second encryption in CCM. After all message blocks have been processed, the output is XORed with the length of the message and associated data, and authenticated just in the last step to produce a tag together with already encrypted nonce.

To find two different plaintexts with the same authentication tag, we choose two consecutive message blocks P_1 and P_2. The formulas for the internal message authentication state are given below:

$$X_1 = (X_0 \oplus C_1) \bullet H,$$
$$X_2 = (X_1 \oplus C_2) \bullet H,$$

where X_0 is the accumulated value from the previous internal message authentication state, and C_1 and C_2 are the ciphertexts produced from the encryption phase.

If we replace the first message block P_1 with an arbitrary new value P_1' then we have:

$$X_1' = (X_0 \oplus C_1') \bullet H.$$

In the next state we want to produce the same authenticated value X_2 but now with the new values X_1' and changed second message block P_2':

$$X_2 = (X_1' \oplus C_2') \bullet H.$$

Thus,

$$X_1 \oplus C_2 = X_1' \oplus C_2',$$

where,

$$C_2' = (X_1 \oplus C_2) \oplus X_1'.$$

So, for P_2' we have:

$$C_2 = P_2 \oplus \mathcal{E}_K(\text{CTR2}),$$
$$C_2' = P_2' \oplus \mathcal{E}_K(\text{CTR2}),$$

where $\text{CTR2} = \text{incr}(\text{CTR1}) = \text{incr}(\text{incr}(N||0^{31}1))$ is a counter, produced from the nonce:

$$P_2' = \mathcal{E}_K(\text{CTR2}) \oplus (X_1 \oplus C_2) \oplus (X_0 \oplus C_1') \bullet H. \tag{3}$$

In Fig. 1 the images a) and c) give the same tag. Note that image a) is the original message, image b) has changed one block in the original message and in the image c) we have used equations (2) or (3) with the values for the second changed block in order to produce the same tag.

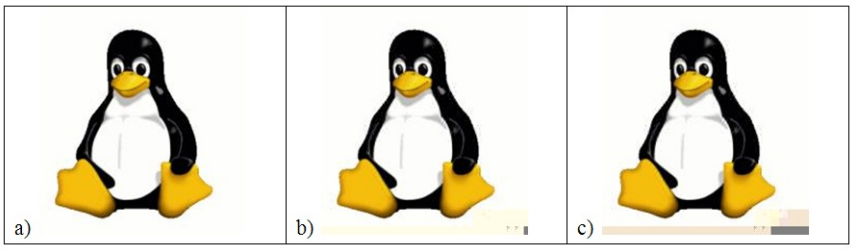


Fig. 1. Simple example of messages with colliding tags for CCM or GCM.

3.3 How to find tag collisions for OCB

OCB3 is the AEAD modification of Offset CodeBook mode [14](OCB). It uses a multiplication in $GF(2^{128})$ but in a simpler way than in GCM. For every message block, each noninitial offset is computed from the prior one by multiplying it by a constant (an operation that has been called *doubling*). OCB3 uses different initial offsets for encryption and authentication phases. For the first one, offset is calculated as a nonce- and key-dependent value, but in the latter it starts from 0. This scheme is on-line: one does not need to know the length of the associated data, the plaintext nor the ciphertext in order to proceed with encryption or decryption.

The tag is produced from the encrypted value of the *Checksum* which is computed as:

$$Checksum = P_1 \oplus \cdots \oplus P_{m-1} \oplus P_m. \tag{4}$$

Since the tag directly depends on the *Checksum* (4) which depends only on the plaintext, we can have the same checksum between two totaly different files. Just the final block of the second file has to be computed to make the checksum the same.

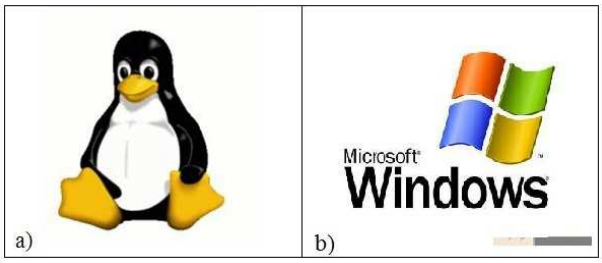


Fig. 2. Simple example of messages with colliding tags for OCB.

In Fig. 2 we can replace the image a) with any image b), and we just compensate the final block of b) in order to produce the same *Checksum* and then the same tag.

4 Conclusion

We have shown that there exist scenarios where it is required that the MAC function retains the properties of a cryptographic hash function, when the key is known. However, the current popular AEAD schemes (such as CCM, GCM, or OCB) do not have this feature. Arguably, protocols and applications built on AEAD schemes having this property will be more robust, which is in accordance with one of the goals of CAESAR. At the forthcoming DIAC 2013 we would like to initiate a discussion about this topic.

References

1. AES. *Advanced Encryption Standard*. FIPS PUB 197, Federal Information Processing Standards Publication, 2001.
2. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. Cryptology ePrint Archive, Report 2000/025, 2000. `http://eprint.iacr.org/`.
3. Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, pages 394–403. IEEE Computer Society, 1997.
4. Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. *IACR Cryptology ePrint Archive*, 2001:35, 2001.
5. D. J. Bernstein. Cryptographic competitions: CAESAR call for submissions, draft 3. May, 21, 2013. Available at `http://competitions.cr.yp.to/caesar-call-3.html`.
6. D. J. Bernstein. Re: secret message numbers. Mailing list of crypto-competitions@googlegroups.com, May 10 2013. `crypto-competitions@googlegroups.com`.
7. T. Krovetz and P. Rogaway. The software performance of authenticated-encryption modes. *Fast Software Encryption - FSE 2011*, 2011.
8. D. McGrew and K. Paterson. Authenticated encryption with aes-cbc and hmac-sha. IETF, Internet-Draft, October 22 2012. `http://tools.ietf.org/html/draft-mcgrew-aead-aes-cbc-hmac-sha2-01`.
9. D. McGrew and J. Viega. *The Galois/Counter Mode of Operation (GCM)*. Natl. Inst. Stand. Technol. `http://www.csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-revised-spec.pdf`.
10. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.
11. C. Mitchell and M. Walker. Solutions to the multidestination secure electronic mail problem. *Computers & Security*, 7:483–488, 1988.
12. Chris J. Mitchell. Multi-destination secure electronic mail. *Comput. J.*, 32(1):13–15, 1989.
13. C. Namprempre, P. Rogaway, and T. Shrimpton. AE5 security notions: Definitions implicit in the CAESAR call. Cryptology ePrint Archive, Report 2013/242, 2013. `http://eprint.iacr.org/`.
14. P. Rogaway, M. Bellare, and J. Black. OCB: A Block-cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.
15. D. Whiting, R. Housley, and N. Ferguson. *Counter with CBC-MAC (CCM)*. Available at `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/`.

Appendix 2

OWCM: One-Way Counter Mode (initial design)

Danilo Gligoroski[1] and Hristina Mihailoska[2] and Håkon Jacobsen[1]

[1] Department of Telematics, Norwegian University of Science and Technology (NTNU), Trondheim, NORWAY, danilog@item.ntnu.no, hakoja@item.ntnu.no
[2] "Ss Cyril and Methodius" University, Faculty of Computer Science and Engineering (FINKI), 1000 Skopje, MACEDONIA, hristina.mihajloska@finki.ukim.mk

We present the initial design of our AEAD scheme based on the counter mode combined with a use of one-way compression functions. Our proposal can be seen as a modification of the GCM scheme [2], where the operations in GHASH are replaced by a use of a double-pipe one-way compression function.

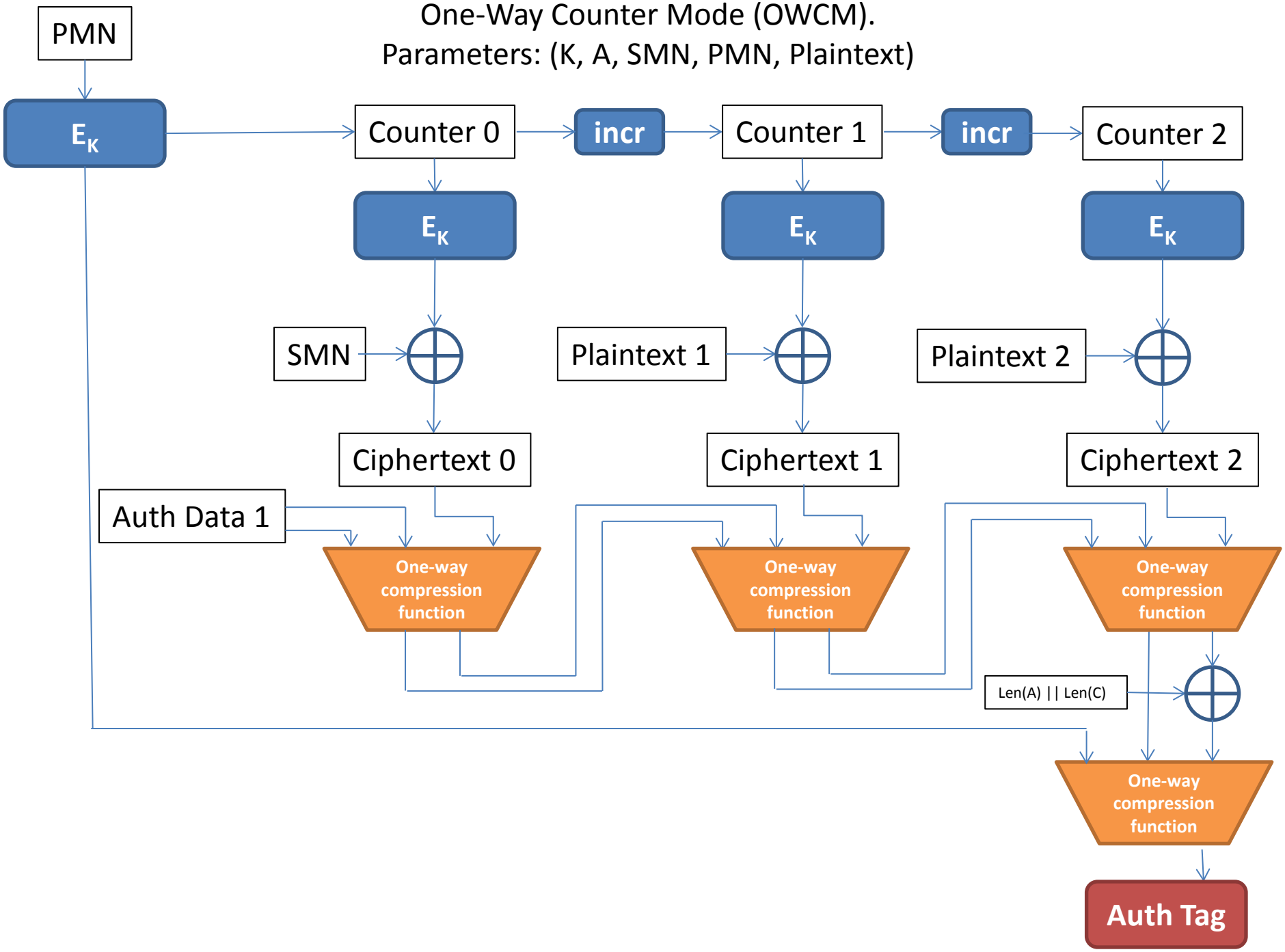
The way how we combine the use of the one-way compression function is similar as that used in the HMAC scheme [1]. Our goal was to design a robust AEAD that will offer the uniqueness of the MAC tags even if the secret key is revealed and to resemble the design style of HMAC that was confirmed as a secure MAC even when the used hash function is not a collision resistant (as long as it is a preimage resistant).

The specific construction of the used one-way function and its efficiency is still under our investigation.

We would like to here comments, critique and suggestions from fellow cryptographers attending DIAC 2013.

References

1. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.
2. D. McGrew and J. Viega. *The Galois/Counter Mode of Operation (GCM)*. Natl. Inst. Stand. Technol. http://www.csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-revised-spec.pdf.


One-Way Counter Mode (OWCM).
Parameters: (K, A, SMN, PMN, Plaintext)
PMN
E_K
Counter 0
incr
Counter 1
incr
Counter 2
E_K
E_K
E_K
SMN
Plaintext 1
Plaintext 2
Ciphertext 0
Ciphertext 1
Ciphertext 2
Auth Data 1
One-way compression function
One-way compression function
One-way compression function
Len(A) || Len(C)
One-way compression function
Auth Tag

One-Way Counter Mode (OWCM).
Parameters: (K, A, SMN, PMN, Plaintext)