Marble Specification

Version 1.2

Designer/Submitter : Jian Guo

Affiliation : Nanyang Technological University, Singapore

Contact Address : ntu.guo@gmail.com

Date : 16 Jan, 2015

Change Log

16/01/2015: the input mask for the last block of associated data is changed from $2^{i-1}3^2L$ to $2^{i-1}3^3L$.

1 Specification

Marble is an authenticated cipher based on reduced AES block ciphers with 128-bit key and 128-bit tag. Nonce is not a necessary input for Marble, and it allows repeated usage of associated data.

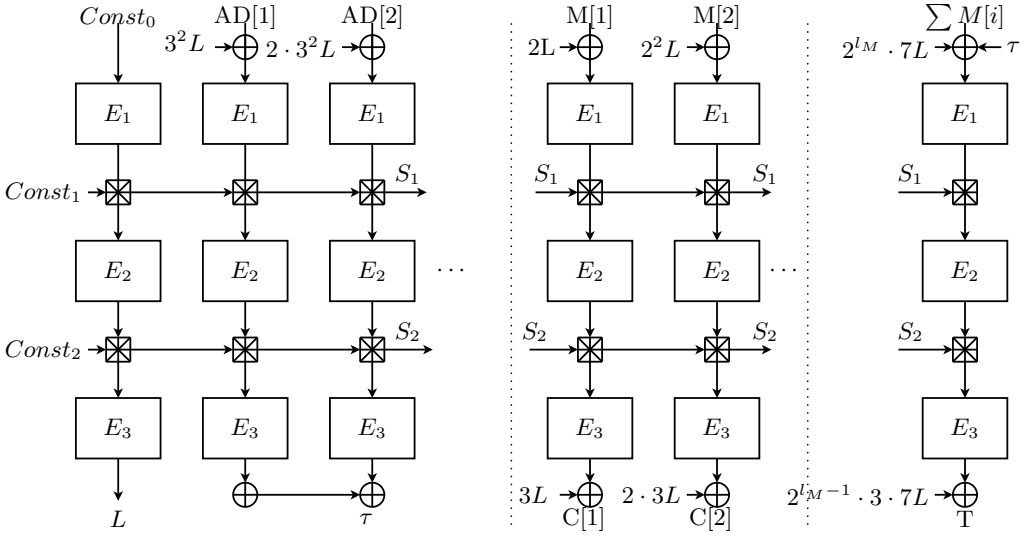
1.1 Notations

In this design, a word in general refers to a 128-bit string, unless otherwise stated. Multiplication is defined in Galois Field (GF) with irreducible polynomial $\texttt{poly}(x) = x^{128} + x^7 + x^2 + x + 1$. We use $X \cdot Y$, or simply XY, to denote GF multiplication of X and Y, and $X + Y$ for bit-wise XOR, depicted as $\bigoplus$ in figures, a function used in the chaining layer $\texttt{TRANS}(x, y) = (x + y, 3x + y)$, depicted as $\boxtimes$. AES round function R consists of SubByte, ShiftRow, MixColumn and AddKey, based on which three block ciphers are defined, each of 4 AES rounds, denoted as E_1, E_2, E_3, with subkeys taken from the first, second and third 4 subkeys of AES-128 key schedule. Note AES-128 generates 11 subkeys with the first key same as the master key, and we use the master key again as the 12-th subkey. There is no whitening key addition for E_1, E_2 and E_3.

A message M, or plaintext is padded when it is less than one block (128 bits), and broken into blocks of 128 bits, $\texttt{M}[1], \texttt{M}[2], \ldots, \texttt{M}[l_M]$ for a message of length in the range $(128 \times (l_M - 1),\ 128 \times l_M]$. The associated data AD is padded always and split into blocks of $\texttt{AD}[1], \texttt{AD}[2], \ldots, \texttt{AD}[l_A]$ for processing. Both padding rules follow the one-zero padding, *i.e.,* a '1' is appended, followed by least number of '0's, so that the total length becomes multiple of blocks.

1.2 Design Description

Fig. 1. Marble design — Overview



An overview of the design is depicted in Fig. 1. In this description, we follow a bottom-up approach to describe the design. The core encryption process is denoted as ENC with pseudo code shown in Algorithm 1. It takes two chaining values S_1 and S_2, two masks MASK1 and MASK2, and the input block IN as input, and outputs a block OUT and updates the two chaining values S_1 and S_2, in short we write (OUT, S_1, S_2) = ENC (IN, S_1, S_2, MASK1, MASK2). Roughly speaking, ENC masks the input IN with MASK1, passes through block cipher call E_1, the output is used to update S_1, followed by block cipher call E_2, then the value is used to update S_2, finally apply block cipher E_3 and mask MASK2. Using the same ENC with different parameters, a mask L is generated and associated data, message and tag are then processed.

Algorithm 1 `ENC`: encryption of a single block

Input: IN, S_1, S_2, MASK1, MASK2
Output: OUT, updated S_1 and S_2

1: ITR = E_1(MASK1 + IN)
2: (S_1, ITR) = `TRANS` (S_1, ITR)
3: ITR = E_2(ITR)
4: (S_2, ITR) = `TRANS` (S_2, ITR)
5: OUT = E_3(ITR) + MASK2

The pseudo code of the workflow is shown in Algorithm 2. Firstly, the mask L is prepared with IN$\leftarrow$ $Const_0$, and preset $S_1 \leftarrow Const_1$ and $S_2 \leftarrow Const_2$, and apply `ENC`, as shown in Algorithm 2 lines 2$\sim$3. This L is then used as the source of masks for the following `ENC` calls. Similar `ENC` is then applied to associated data and message blocks with distinct masks, *i.e.,* input mask $2^{i-1}3^2L$ for the i-th block of associated data for $i = 1, \ldots, l_A - 1$ and $2^{i-1}3^3L$ for the last block $i = l_A$. No output mask is used for associated data processing. 2^iL and $2^{i-1} \cdot 3L$ are used as input/output masks for i-th message block. Lastly, $2^{l_M} \cdot 7L$ and $2^{l_M-1}3 \cdot 7L$ are used as input/output mask for tag generation. For simplicity we set $Const_0 = 0$, $Const_1 = 1$ and $Const_2 = 2$.

Algorithm 2 Marble: pseudocode for the workflow, with multiple blocks of `M`

Input: `AD`, `M`, `K`
Output: `C`, `T`

1: Initialize the subkeys for E_1, E_2 and E_3 with the master key `K`
2: $S_1 = Const_1$, $S_2 = Const_2$, $\tau = 0$, $S_M = 0$
3: (L, S_1, S_2) = `ENC` ($Const_0$, S_1, S_2, 0, 0)
4: Apply one-zero padding to `AD`, and break `M` and padded `AD` into blocks
5: for $i = 1, \ldots, l_A$
6: **if** $i \neq l_A$ **then**
7: (OUT, S_1, S_2) = `ENC` (`AD`[i], S_1, S_2, $2^{i-1} \cdot 3^2L$, 0)
8: **else**
9: (OUT, S_1, S_2) = `ENC` (`AD`[i], S_1, S_2, $2^{i-1} \cdot 3^3L$, 0)
10: **end if**
11: $\tau = \tau + \mathrm{OUT}$
12: endfor
13: for $i = 1, \ldots, l_M$
14: (`C`[i], S_1, S_2) = `ENC` (`M`[i], S_1, S_2, $2^i \cdot L$, $2^{i-1} \cdot 3L$)
15: $S_M = S_M +$ `M`[i]
16: endfor
17: (`T`, S_1, S_2) = `ENC` ($S_M + \tau$, S_1, S_2, $2^{l_M} \cdot 7L$, $2^{l_M-1} \cdot 3 \cdot 7L$)

1.3 Processing non-full block message

For non-full block message, tag splitting technique [4] is used when the message is less than one block, and XLS [9] technique otherwise.

Tag splitting for $l = |M| < 128$. Firstly, define $(\mathtt{C}', \mathtt{T}')$ by processing the padded message block as usual

$(\mathtt{C}', \mathtt{T}') \leftarrow \mathsf{Marble}(M\|10^*)$ with block index $l_M = 0$, and input/output masks of message block are L and $2^{-1}3L$.
$\mathtt{C} \leftarrow \lceil \mathtt{C}' \rceil_l$ leftmost l bits
$\mathtt{T} \leftarrow \lfloor \mathtt{C}' \rfloor_{128-l} \| \lceil \mathtt{T}' \rceil_l$ rightmost $128 - l$ bits of $\mathtt{C}'$ concatenated with leftmost l bits of $\mathtt{T}'$

XLS for non-full message block with $l > 128$. Following COPA [2], XLS processes one full block (tag of the previous $l_M - 1$ full blocks) + one last partial block, and output a ciphertext with equal length of the partial plaintext block and tag by making three calls to `ENC`. One can then process the first $l_M - 1$

full message blocks as usual, and last non-full block is processed by XLS. Note the chaining values S_1 and S_2 are not updated in the XLS process, in order to allow decryption.

1.4 Decryption Algorithm

Decryption can be defined by reversing ENC for the plaintext/ciphertext part only, and the workflow is exactly the same as for encryption, except replacing ENC by DEC. The details of DEC is shown in Algorithm 3, where $E_1^{-1}, E_2^{-1}, E_3^{-1}$ are defined similarly as for AES decryption, and $\texttt{TRANS}^{-1}(x, y) = (2x + y, 3x + y)$.

Algorithm 3 DEC: decryption of a single block

Input: C, S_1, S_2, MASK1, MASK2
Output: M, updated S_1 and S_2

1: ITR = E_3^{-1}(MASK2 + C)
2: (S_2, ITR) = $\texttt{TRANS}^{-1}$(S_2, ITR)
3: ITR = E_2^{-1}(ITR)
4: (S_1, ITR) = $\texttt{TRANS}^{-1}$(S_1, ITR)
5: M = E_1^{-1}(ITR) + MASK1

1.5 Recommended Parameters

KEY SIZE	ASSOCIATED DATA	SECRET MESSAGE NUMBER	PUBLIC MESSAGE NUMBER	PLAINTEXT LENGTH
128	unlimited	not supported	not supported	unlimited

As above, there is only one recommended parameter for Marble, which takes input of a key of 128 bits, no nonce, an associated data with no length limit (it can be null string), a message (or plaintext) with no length limit (it can be null string) with restriction that associated data and message cannot be empty at the same time. The tag size is 128 bits. Ciphertext length (excluding tag) is same as plaintext length. Marble does not support *public message number* or *private message number* in the recommended parameter. However, one can opt to use one block of public message number (or nonce) by replacing $Const_0$ with it, and one block of private message number as first block of message, and these two numbers must always be present once this option is chosen, not to use interactively with the main recommended parameters since these two versions are mutual exclusive. Again, this option is not in our recommendation.

2 Security Goals

The security goals of Marble are to achieve full security level for both privacy and authenticity, for both cases when the associated data is unique or re-used. Marble is also decryption mis-use resistant, *i.e.,* attacker is allowed to access the decryption oracle without checking the integrity of the tag. We don't claim resistance against any distinguisher or any security in related-key settings, or any other security against attacks rather than forgery or key recovery. Since the design is optimized for speed with minimum number of rounds required for the block ciphers E_1, E_2 and E_3, hence we don't claim security with reduced-round ciphers either.

3 Security Analysis

The security of Marble is not proved, hence we show our arguments against recent attacks, including here slide attack, generalized birthday attack and differential attack. Generally speaking, ENC consists of 12 rounds, 2 more rounds than AES-128, hence we expect higher security margin against attacks involving single block plaintext/ciphertext. Hence, here we focus on attacks involving more than one plaintext/ciphertext blocks.

Table 1. Summary of claimed security of Marble in bits, including associated data re-using scenario, '-' means security not claimed.

Property	unique AD	AD re-use
Confidentiality for the plaintext	128	128
Integrity for the plaintext	128	128
Integrity for the associated data	128	-
Decryption mis-use	128	128
Related-key	-	-
Distinguisher	-	-

3.1 Slide Attack

It is proved in [2] that the masks will not collide, hence difficult to launch slide attack between ENC calls of message blocks at different block positions. It is also difficult to slide between the AES rounds inside one ENC call, due to the different constants used in the AES-128 subkey generation.

3.2 Generalized Birthday Attacks

We note that COPA [2] and POET [1, 7] are not decryption resistant due to the fact that a block difference in ciphertext affects only a limited number of plaintext blocks. In particular, a difference in ciphertext block affects two plaintext blocks of COPA, and hence one can generate many lists by choosing different positions of the ciphertext block independently and generalized birthday attack applies. This is not the case for Marble due to the choice of TRANS function, which ensures at least three out of 4 values (2 inputs and 2 outputs) will consist differences. Difference in a single block will be propagated further to later blocks through the chaining values S_1 and S_2. One might think of two or more blocks so that the chaining value collides, this does not apply either due to the reason below.

3.3 Differential Attacks

In case when the difference appears in more than one blocks, any differential characteristic or differential will involve at least 8 AES rounds, which is not vulnerable to any key recovery type of attacks, as far as we are aware of. Furthermore, a distinguishing property in either S_1 or S_2 is difficult to detect/validate. For instance, a collision of S_1 cannot be easily verified, since this value is "masked" by E_1^{-1} in decryption oracle or $E_3 \circ E_2$ in encryption oracle. A zero difference in both S_1 and S_2 is detectable, however, this has to involve at least 16 AES rounds in total in a differential resulting this. It is noted that maximum expected differential probability is about 2^{-112} [3], one cannot fulfil two 4-round differentials independently in this design due to the chaining values.

4 Features

4.1 Features and Use Cases

Marble is online, i.e., a message in current block will affect the value of current and all subsequent ciphertext blocks and tag values. Marble aims to be robust, so that it can be used in many extreme use cases such as:

1. Encryption/Decryption only: Marble aims to achieve confidentiality without the tag, by opting out the tag generation part.
2. Integrity of associated data: Marble allows empty message, and takes only associated data and outputs the tag T. This is processed as usual, but opt out the message processing part, and set $\sum \mathtt{M}[i] = 0$ for tag generation. Note, in this case, the input mask for tag generation is set uniquely to $7L$ with $l_M = 0$. This can be used to check the integrity of associated data.
3. MAC only: Marble is also allowed to achieve integrity alone, by showing the plaintext together with the tag, which is generated as usual but discard the ciphertext.

4. Empty associated data: Marble is allowed to take no associated data, by processing a length "0" associated data which will be padded following one-zero padding as usual, for both authenticity and integrity functionalities.
5. Other inputs, those requiring confidentiality (such as secret message number), can be treated as part of the message, and those requiring only integrity (such as public message number) as part of associated data.

4.2 Software speed

A preliminary implementation result shows that Marble runs at 1.6 cycles per byte on an Intel(R) Core(TM) i5-4570 CPU (Haswell micro-architecture) clocked at 3.20 GHz with AES new instruction for a message of 8 KBytes.

4.3 Comparison with AES-GCM

Advantage over AES-GCM. Besides being online, Marble is designed with robustness in mind, and claims to be nonce mis-use and decryption mis-use resistant, with full security levels under both scenarios. Yet the software speed with AES-NI is comparable with AES-GCM [6] on high end CPUs, and believed to be faster on other low end CPUs. Marble requires doubling in Galois Field only, while AES-GCM requires full GF multiplication on 128 bits.

Disadvantage. Marble has no security proof so far, but this does not eliminate the possibility proving the security at certain bound.

Parameter set consideration. The parameters of Marble is chosen in the way that it achieves the security claimed with minimum number of rounds required, this is to achieve maximum speed. Due to robustness of Marble, there is merely no restriction on the length of varies inputs.

5 Design Rationale

The designer has not hidden any weaknesses in this cipher.

5.1 $2n$-bit chaining

In order to achieve full security level in case of nonce/AD reuse, this "chaining" needs to have at least $2n$ bits, otherwise, a birthday attack to find colliding "chaining" values applies, this is the key reason that we have $2n$ bit chaining in total. This strategy has been noted and widely adopted by hash function designs since [8].

5.2 Parallelization and being online

For an authenticated cipher being online and parallel processing seems contradicting properties for an AE, some "chaining" value has to be passed to next block for it to be online, so this has to be done in sequential. In the meanwhile, being parallel requires processing message blocks independently as much as possible. Hence, "chaining" computation has to be fast in software, otherwise it can become the bottleneck of the software speed, as for [1]. Hence, we choose the simple $\texttt{TRANS}(x, y) = (x + y, 3x + y)$, which involves one doubling and three xor operations. Furthermore, to update the chaining S_1 and S_2, there is no doubling involved, and requires only a single XOR operation, which can be realized fast.

5.3 Masking and pre-/post- processing

It is in general not a good idea to reveal internal state/chaining when one wants to claim AD misuse resistance. To protect the two chaining values S_1 and S_2, pre-process the message block using E_1 and a post-processing of E_3 for ciphertext is necessary. An additional minimum process between the two chaining values adds up the three-layer construction. A layer of masking for plaintext/ciphertext adds more to the pre-/post- processing, besides resisting slide attacks.

6 Intellectual Property

This design is patent free, and the designer has no intend to file a patent. If any of this information changes, the submitter will promptly (and within at most one month) announce these changes on the crypto-competitions mailing list.

7 Consent

The submitter hereby consents to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitter understands that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analyses that led to the selection of the algorithm. The submitter understands that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitter acknowledges that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitter understands that if they disagree with published analyses then they are expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitter understands that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

Acknowledgements. We thank Jérémy Jean, Thomas Peyrin and Lei Wang for fruitful discussions, and Tetsu Iwata for introducing me to the area of authenticated encryption. The work in this design was supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06). We thank Thomas Thur, Gaëtan Leurent, and Valentin Suder for their analysis work [5] pointing out a weakness in the input mask version 1.1.

References

1. Farzaneh Abed, Scott Fluhrer, John Foley, Christian Forler, Eik List, Stefan Lucks, David McGrew, and Jakob Wenzel. Pipelineable On-Line Encryption. In *FSE*, 2014, preproceedings version.
2. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser, and Kan Yasuda. Parallelizable and Authenticated Online Ciphers. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (1)*, volume 8269 of *LNCS*, pages 424–443. Springer, 2013.
3. Joan Daemen, Mario Lamberger, Norbert Pramstaller, Vincent Rijmen, and Frederik Vercauteren. Computational aspects of the expected differential probability of 4-round AES and AES-like ciphers. *Computing*, 85(1-2):85–104, 2009.
4. Ewan Fleischmann, Christian Forler, and Stefan Lucks. McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In Anne Canteaut, editor, *FSE*, volume 7549 of *LNCS*, pages 196–215. Springer, 2012.
5. Thomas Fuhr, Gaëtan Leurent, and Valentin Suder. Forgery and Key-Recovery Attacks on CAESAR Candidate Marble. https://hal.inria.fr/hal-01102031, January 2015.
6. Shay Gueron. AES-GCM software performance on the current high end CPUs as a performance baseline for CAESAR competition. Presented at DIAC 2013 workshop, slides available http://2013.diac.cr.yp.to/slides/gueron.pdf, 2013.
7. Jian Guo, Jérémy Jean, Thomas Peyrin, and Lei Wang. Breaking POET Authentication with a Single Query. Cryptology ePrint Archive, Report 2014/197, 2014. http://eprint.iacr.org/.
8. Stefan Lucks. A Failure-Friendly Design Principle for Hash Functions. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *LNCS*, pages 474–494. Springer, 2005.
9. Thomas Ristenpart and Phillip Rogaway. How to Enrich the Message Space of a Cipher. In Alex Biryukov, editor, *FSE*, volume 4593 of *LNCS*, pages 101–118. Springer, 2007.