Marble Specification Version 1.0

Designer/Submitter : Jian Guo

Affiliation : Nanyang Technological University, Singapore

Contact Address : guojian@ntu.edu.sg

Submission Date : March 15, 2014

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.

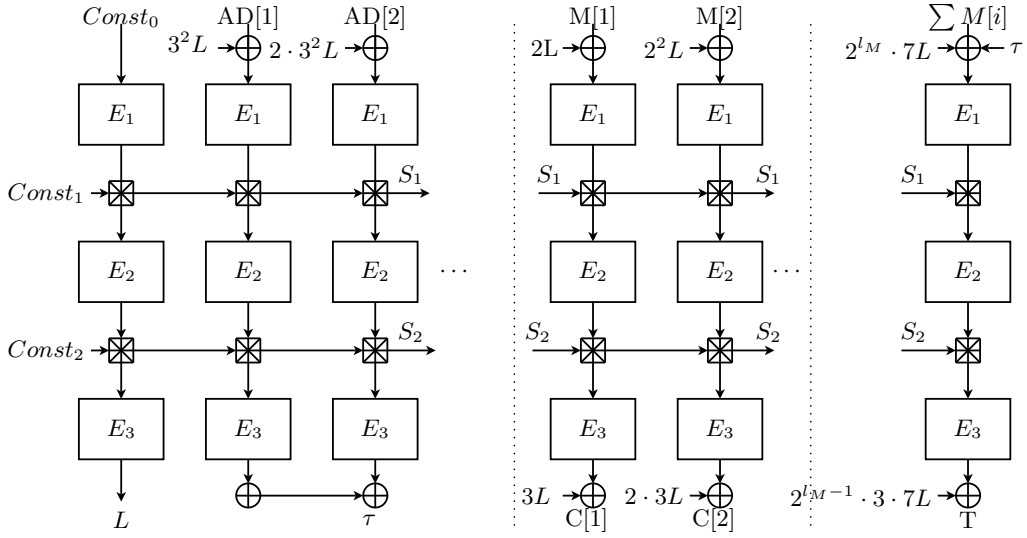
1.1 Notations

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 ⊠. 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 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 $(128 \times (l_M - 1), 128 \times l_M]$. Similarly, the associated data AD is split into blocks of $\texttt{AD}[1], \texttt{AD}[2], \ldots, \texttt{AD}[l_A]$ for processing.

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∼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 $2^{l_A-1}3^3L$ for the last block when it is full block, and $2^{l_A-1}3^4L$ for last partial block. 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` and `AD`

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: for $i = 1, \ldots, l_A$
5: (OUT, S_1, S_2) = `ENC` (`AD`[i], S_1, S_2, $2^{i-1} \cdot 3^{2+\lfloor i/l_A \rfloor}L$, 0)
6: $\tau = \tau$ + OUT
7: endfor
8: for $i = 1, \ldots, l_M$
9: (`C`[i], S_1, S_2) = `ENC` (`M`[i], S_1, S_2, $2^i \cdot L$, $2^{i-1} \cdot 3L$)
10: $S_M = S_M +$ `M`[i]
11: endfor
12: (`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 the last block of associated data

When the last block of associated data is of full block, the input mask is $2^{l_A-1} \cdot 3^3L$, distinguished from $2^i \cdot 3^2L$ as for other blocks, as depicted in the left part of Fig. 2. Otherwise, the last block is of length less than 128 bits, a '1' bit is padded, followed by least number of '0's so that it becomes a full block, the input mask $2^{l_A-1} \cdot 3^4L$ is applied, as depicted in the right part of Fig. 2.

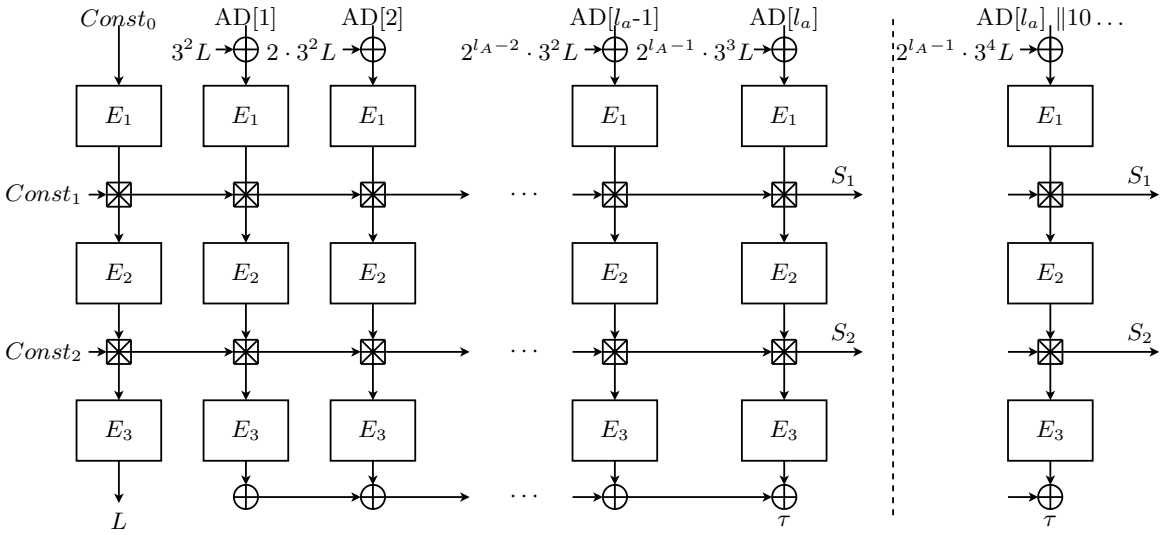
1.4 Processing non-full block message

For non-full block message, tag splitting technique [4] is used when the message is less than one block, and XLS [7] technique otherwise.

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

$(\mathtt{C}', \mathtt{T}') \leftarrow \mathsf{Marble}(M \| 10^*)$
$\mathtt{C} \leftarrow \lceil \mathtt{C}' \rceil_l$ 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 $\mathtt{T}'$

Fig. 2. Marble design — associated data processing



XLS for non-full message block with $l > 128$. XLS processes one full block + one partial block, and output a ciphertext with equal length by making three calls to `ENC`. One can then process the first $l_M - 2$ full message blocks as usual, and last two blocks (inclusive of the last partial block) are processed by XLS, followed by tag generation. Note the chaining values S_1 and S_2 remain unchanged in the first two `ENC` calls and are only updated in the last `ENC` call of the XLS, in order to allow decryption.

1.5 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

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.

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.

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, 5] 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 fulfill 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:

- Encryption/Decryption only: Marble aims to achieve confidentiality without the tag, by opting out the tag generation part.
- 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 $2 \cdot 7L$ with $l_M = 0$. This can be used to check the integrity of associated data.
- 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.
- Empty associated data: Marble is allowed to take no associated data, by opting out the associated data processing part and setting $\tau = 0$ for tag generation, for both authenticity and integrity functionalities.

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 clocked at 3.20 GHz with AES new instruction for a message of 8 KBytes.

4.3 Other parameters

The current proposal is for maximum software performance, one may choose to use 6-round E_1, E_2, E_3 and $\texttt{TRANS}(x, y) = (3x + y, x + 3y)$ for better security. Furthermore, one may consider using `AES` itself to replace E_1, E_2, E_3 for even larger security margin.

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 [6].

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 udpate 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. The work in this design was supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

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. 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/.
6. 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.
7. 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.