AES-OTR v3.1

Designer/Submitter: Kazuhiko Minematsu (NEC Corporation, Japan)

Contact Address: k-minematsu@ah.jp.nec.com

Date: September 1, 2016

1 Specification

OTR, which stands for Offset Two-Round, is a blockcipher mode of operation to realize an authenticated encryption with associated data (AEAD), proposed by Minematsu [25, 26]. Our CAESAR submission, AES-OTR, is an instantiation of OTR based on AES blockcipher with an additional version for associated data processing.

1.1 Parameters

AES-OTR has the following four parameters.

- Key length : 16 bytes (128 bits), 24 bytes (192 bits), 32 bytes (256 bits).
- Nonce length : 1 byte (8 bits) to 15 bytes (120 bits).
- Tag length : 4 bytes (32 bits) to 16 bytes (128 bits).
- Associated Data Processing (ADP) : Parallel (p) or Serial (s).

1.2 Recommended Parameter Set

- Primary recommended parameter set `aes128otrpv1` : 16-byte key for AES-128, 12-byte nonce, 16-byte tag, and parallel ADP.
- Secondary recommended parameter set `aes128otrsv1` : 16-byte key for AES-128, 12-byte nonce, 16-byte tag, and serial ADP.
- Third recommended parameter set `aes256otrpv1` : 32-byte key for AES-256, 12-byte nonce, 16-byte tag, and parallel ADP.
- Fourth recommended parameter set `aes256otrsv1` : 32-byte key for AES-256, 12-byte nonce, 16-byte tag, and serial ADP.

All parameters are targeted for Use Case 2 (High-performance applications). They will also perform well on some (but not all) lightweight applications.

1.3 Algorithm

■Basic Notations. Let $\{0,1\}^*$ and $\{0,1\}^{8*}$ be the sets of all finite-length binary strings and all finite-length byte strings, both including the empty string ε. Here $\{0,1\}^{8*} \subset \{0,1\}^*$. For binary string $X \in \{0,1\}^*$, its bit length is denoted by $|X|$. We define $|X|_a \stackrel{\text{def}}{=} \max\{\lceil |X|/a \rceil, 1\}$. For $X = \varepsilon$ we have $|X|_a = 1$ for any $a \geq 1$ and $|X| = 0$. For any $X, Y \in \{0,1\}^*$ we write $X\|Y$ or simply XY to denote the concatenation of X and Y. A sequence of a zeros is denoted by 0^a. For $X \in \{0,1\}^*$, we write $(X[1], \ldots, X[x]) \xleftarrow{n} X$ to denote the n-bit block partitioning of X. That is, $X[1]\|X[2]\|\ldots\|X[x] = X$ where $x = |X|_n$, and $|X[i]| = n$ for $i < x$ and $|X[x]| \leq n$. If $X = \varepsilon$ we have $X[1] \xleftarrow{n} X$ with $X[1] = \varepsilon$, for any $n \geq 1$. The sequence of first c bits of $X \in \{0,1\}^*$ is denoted by $\mathtt{msb}_c(X)$. We have $\mathtt{msb}_0(X) = \varepsilon$ for any X. For $X \in \{0,1\}^*$ with $0 \leq |X| < n$, $\underline{X}$ denotes the 10^* padding written as $X\|10^{n-|X|-1}$. When $|X| = n$, we have $\underline{X} = X$, and when $X = \varepsilon$ we have $\underline{X} = 10^{n-1}$.

Unless otherwise stated, we assume a variable n to mean 128. We also assume E_K to mean AES encryption function with key K, for some fixed $|K| \in \{128, 192, 256\}$. For $X, Y \in \{0,1\}^n$, $Y = E_K(X)$ means that Y is the ciphertext obtained by AES encryption for plaintext X with key K. We may simply write E to denote E_K. When we write $\Pr[\mathsf{X} : \mathsf{Y}]$, it means the probability of event Y defined over the experiment X. We may omit X if it is obvious. If adversary $\mathcal{A}$ accesses an oracle O and returns a value x as a final output, we write $\mathcal{A}^O \Rightarrow x$ to denote the corresponding event. For non-negative integer x, let $\mathtt{n2s}(x, n)$ (number to string) denote the standard n-bit encode of x. For example, $\mathtt{n2s}(11, 5) = 01011$.

The specification of our scheme assumes big endian.

■Doubling. We may view $X \in \{0,1\}^n$ as a coefficient vector of the polynomial of $\mathrm{GF}(2^n)$. Following Rogaway [29], by writing $2X$ we mean the multiplication of the generator of the field $\mathrm{GF}(2^n)$ (i.e. polynomial $\mathtt{x}$) and X over $\mathrm{GF}(2^n)$, which is called *doubling*. We write 2^iX to denote i-time doublings of X, and $4X$ to denote $2^2X = 2(2X)$, and $3X$ to denote $2X \oplus X$, and $7X$ to denote $2(2X) \oplus 2X \oplus X$. A multiplication over $\mathrm{GF}(2^n)$ is done by multiplying two input polynomials, and dividing by an irreducible polynomial, $f(\mathtt{x})$, and taking the remainder. For $n = 128$, we fix $f(\mathtt{x}) = \mathtt{x}^{127} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x}^1 + 1$. In this case, as shown by [19] $2X$ is implemented as

$$2X = \begin{cases} X \ll 1 & \text{if } \mathtt{msb}_1(X)\text{=}0 \\ (X \ll 1) \oplus 0^{120}10000111 & \text{if } \mathtt{msb}_1(X)\text{=}1 \end{cases} \tag{1}$$

where $X \ll 1$ denotes 1-bit logical left shift of X.

■Input and Output. The encryption algorithm of AES-OTR using blockcipher (here AES) E, tag bit length τ, with parallel ADP is written as OTR-$\mathcal{E}_{E,\tau,p}$. When ADP is serial we similarly write OTR-$\mathcal{E}_{E,\tau,s}$ to denote the encryption algorithm.

Both algorithms take the following byte sequences.

- Key $K \in \mathcal{K}_{ae}$,
- Nonce (a public message number) $N \in \mathcal{N}_{ae}$,
- Associated Data (AD) $A \in \mathcal{A}_{ae}$, and
- Plaintext $M \in \mathcal{M}_{ae}$.

The output is a byte sequence

- Ciphertext $(C, T) \in \mathcal{M}_{ae} \times \mathcal{T}_{ae}$,

where a ciphertext is a concatenation of unauthenticated ciphertext C fulfilling $|C| = |M|$ and a tag $T \in \mathcal{T}_{ae}$. There is no secret message number, hence the length of secret message number is assumed to be zero. Here $\mathcal{N}_{ae} = \{0,1\}^{|N|}$, $\mathcal{K}_{ae} = \{0,1\}^{|K|}$ with fixed $|N|$ and $|K|$ satisfying $|N| \in \{8, 16, \ldots, 120\}$ and $|K| \in \{128, 192, 256\}$, and $\mathcal{M}_{ae} = \mathcal{A}_{ae} = \{0,1\}^{8*}$, but we restrict these lengths as $|M|_8 \leq 2^{64}$ and $|A|_8 \leq 2^{64}$ (so 2^{60} 16-byte blocks). Both M and A can be empty (possibly simultaneously, though artificial). Tag space $\mathcal{T}_{ae}$ is $\{0,1\}^\tau$ with fixed $\tau \in \{32, 40, \ldots, 128\}$.

The decryption algorithm of AES-OTR using blockcipher E, and tag bit length τ, with parallel ADP is written as OTR-$\mathcal{D}_{E,\tau,p}$. Similarly we write OTR-$\mathcal{D}_{E,\tau,s}$ to denote the decryption algorithm with serial ADP. Both algorithms take the following byte sequences.

- Key $K \in \mathcal{K}_{ae}$,
- Nonce (a public message number) $N \in \mathcal{N}_{ae}$,
- Associated Data (AD) $A \in \mathcal{A}_{ae}$, and
- Ciphertext $(C, T) \in \mathcal{M}_{ae} \times \mathcal{T}_{ae}$,

where a ciphertext is a concatenation of unauthenticated ciphertext C and a tag $T \in \mathcal{T}_{ae}$. If the result is determined as authentic, the output is a byte sequence

- Plaintext $M \in \mathcal{M}_{ae}$ satisfying $|M| = |C|$,

otherwise a rejection symbol $\perp$.

We remark that parallel ADP version is identical to the algorithm shown by [25] except tag-length encoding in nonce processing.

■Algorithms for Parallel ADP. The algorithms of OTR-$\mathcal{E}_{E,\tau,p}$ and OTR-$\mathcal{D}_{E,\tau,p}$ are described in Fig. 1, and OTR-$\mathcal{E}_{E,\tau,p}$ is illustrated in Fig 2. In Fig. 1 the functions OTR-$\mathcal{E}_{E,\tau,p}$ and OTR-$\mathcal{D}_{E,\tau,p}$ are further decomposed into the encryption and decryption cores, $\mathrm{EF}_{E,\tau}$, $\mathrm{DF}_{E,\tau}$, and the authentication core, AF_E. Here tag bit length τ is a fixed parameter and we may simply write as EF_E, DF_E. Below we provide a brief explanation on Fig. 1. For OTR-$\mathcal{E}_{E,\tau,p}$, we first partition a plaintext M into n-bit blocks, i.e. $(M[1], \ldots, M[m]) \xleftarrow{n} M$ (Fig. 1, Line 3 of EF_E). For presentation purpose we also assume a partition

into $2n$-bit blocks, which we call chunks, denoted as $(\boldsymbol{M}[1], \ldots, \boldsymbol{M}[\ell]) \xleftarrow{2n} M$. For each $i < \ell$, the i-th chunk $\boldsymbol{M}[i] = (M[2i-1], M[2i])$ is encrypted by a two-round Feistel permutation with masks as

$$C[2i-1] = E_K(2^{i-1}L \oplus M[2i-1]) \oplus M[2i], \tag{2}$$

$$C[2i] = E_K(2^{i-1}3L \oplus C[2i-1]) \oplus M[2i-1], \tag{3}$$

where L is an encrypted nonce with tag-length (τ) encoding, written as $E_K(\mathrm{Format}(\tau, N))$ (see below). With these computations we obtain the i-th ciphertext chunk, $\boldsymbol{C}[i] = (C[2i-1], C[2i])$. For the last chunk (which may be partial), a variation of the above procedure is applied depending on the length of the last chunk. When it is more than n bits we apply a variant of two-round Feistel, and otherwise a variant of CTR mode is applied. This yields the unauthenticated ciphertext, $C = (\boldsymbol{C}[1], \ldots, \boldsymbol{C}[\ell])$. For computing the tag, the check sum is computed by taking the sum (XOR) of right halves of the plaintext chunks, $\boldsymbol{M}[1], \ldots, \boldsymbol{M}[\ell-1]$, i.e., even plaintext blocks. The check sum also involves a value from the final chunk $\boldsymbol{M}[\ell]$ which depends on the chunk length. Then the check sum is given to the AES encryption with a dedicated mask, depending on the length of last chunk. The result is the plaintext tag before truncation, TE.

In parallel to the above procedure, an associated data A is given to AF_E. It uses a parallelizable PRF, which is a variant of PMAC [29]. Then we obtain the untrancated associated data tag, TA. This is when A is non-empty. When $A = \varepsilon$, we have $TA = 0^n$.

Finally, we take a sum of TE and TA and apply the truncation to obtain the tag T. By concatenating C and T we obtain the ciphertext (C, T).

For OTR-$\mathcal{D}_{E,\tau,p}$, we first perform the inverse operation of (2) and (3) and the inverse operation for the last chunk, to obtain the (unverified) plaintext $M = (\boldsymbol{M}[1], \ldots, \boldsymbol{M}[\ell])$, and compute the check sum to obtain TE. By computing TA and taking a sum with TE, we obtain the unverified tag $\widehat{T}$. If $T = \widehat{T}$ we decide ciphertext (C, T) is authentic and the final output is M. Otherwise, the output is $\perp$.

The encoding of nonce with tag length, $\mathrm{Format}(\tau, N)$, is the same as that used in OCB [21], including the fact that $|N| \leq 120$ when $n = 128$.

■Algorithms for Serial ADP. For serial ADP, the procedures are very similar to the case of parallel ADP, except the handling of associated data. We have Fig. 3 to show the algorithms of OTR-$\mathcal{E}_{E,\tau,s}$ and OTR-$\mathcal{D}_{E,\tau,s}$, and OTR-$\mathcal{E}_{E,\tau,s}$ is also illustrated in Fig 4. The algorithms of Fig. 3 are further decomposed into encryption core EF-$\mathrm{S}_{E,\tau}$, decryption core DF-$\mathrm{S}_{E,\tau}$ (where we may omit τ from them for simplicity), and authentication core AF-S_E. For EF-S_E and DF-S_E the pseudo-codes are the same as EF_E and DF_E except the generation of U (line 2). That is, we first take a sum of TA and $E_K(\mathrm{Format}(\tau, N))$ and U is a doubling of it. Here TA is generated by AF-S_E taking A and AF-S_E is a variant of CMAC [5], also known as OMAC [19].

2 Security Goals

We consider the standard security notions for nonce-based AEs [30], see Section 3. Nonce (public message number) must be unique for all encryption queries, however the same nonce can be used for an encryption and a decryption query, or for two decryption queries. We basically do not claim any security when nonce is reused for encryption queries, though the serial ADP version slightly relaxes the condition (See the last of this section). We do not claim security when a key is used across different parameters, e.g. when OTR with parallel ADP and serial ADP are concurrently used with one key. This specification assumes that the tag length is fixed in use, which follows CAESAR call for submissions. The nonce encoding with tag length is for clarity, compatibility with OCB, and preventing trivial misuse discussed at [1]. See also [17, 32].

The security goal of our cipher for confidentiality (or privacy) of plaintext is shown in Table 1, and that for integrity (or authenticity) of plaintext, associated data, and public message number is shown in Table 2. We believe that, the security goal of our cipher is mostly explained by the security bounds presented in Section 3. These tables come from a natural interpretation of these bounds, where the variables in the tables denote the required workload of an adversary to break the cipher, in logarithm

Algorithm OTR-$\mathcal{E}_{E,\tau,p}(N, A, M)$

1. $(C, TE) \leftarrow \mathrm{EF}_{E,\tau}(N, M)$
2. **if** $A \neq \varepsilon$ **then** $TA \leftarrow \mathrm{AF}_E(A)$
3. **else** $TA \leftarrow 0^n$
4. $T \leftarrow \mathtt{msb}_\tau(TE \oplus TA)$
5. **return** (C, T)

Algorithm OTR-$\mathcal{D}_{E,\tau,p}(N, A, C, T)$

1. $(M, TE) \leftarrow \mathrm{DF}_{E,\tau}(N, C)$
2. **if** $A \neq \varepsilon$ **then** $TA \leftarrow \mathrm{AF}_E(A)$
3. **else** $TA \leftarrow 0^n$
4. $\widehat{T} \leftarrow \mathtt{msb}_\tau(TE \oplus TA)$
5. **if** $\widehat{T} = T$ **return** M
6. **else return** $\perp$

Algorithm $\mathrm{EF}_{E,\tau}(N, M)$

1. $\Sigma \leftarrow 0^n$
2. $U \leftarrow E(\mathrm{Format}(\tau, N))$, $L \leftarrow U$, $L^\sharp \leftarrow 3U$
3. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
4. **for** $i = 1$ **to** $\lceil m/2 \rceil - 1$ **do**
5. $\quad C[2i-1] \leftarrow E(L \oplus M[2i-1]) \oplus M[2i]$
6. $\quad C[2i] \leftarrow E(L^\sharp \oplus C[2i-1]) \oplus M[2i-1]$
7. $\quad \Sigma \leftarrow \Sigma \oplus M[2i]$
8. $\quad L \leftarrow L \oplus L^\sharp$, $L^\sharp \leftarrow 2L^\sharp$ // $L = 2^i U$, $L^\sharp = 2^i 3U$
9. **if** m **is even**
10. $\quad Z \leftarrow E(L \oplus M[m-1])$
11. $\quad C[m] \leftarrow \mathtt{msb}_{|M[m]|}(Z) \oplus M[m]$
12. $\quad C[m-1] \leftarrow E(L^\sharp \oplus \underline{C[m]}) \oplus M[m-1]$
13. $\quad \Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$
14. $\quad L^* \leftarrow L^\sharp$
15. **if** m **is odd**
16. $\quad C[m] \leftarrow \mathtt{msb}_{|M[m]|}(E(L)) \oplus M[m]$
17. $\quad \Sigma \leftarrow \Sigma \oplus \underline{M[m]}$
18. $\quad L^* \leftarrow L$
19. **if** $|M[m]| \neq n$ **then** $TE \leftarrow E(3^2 L^* \oplus \Sigma)$
20. **else** $TE \leftarrow E(7L^* \oplus \Sigma)$
21. $C \leftarrow (C[1], \ldots, C[m])$
22. **return** (C, TE)

Algorithm $\mathrm{DF}_{E,\tau}(N, C)$

1. $\Sigma \leftarrow 0^n$
2. $U \leftarrow E(\mathrm{Format}(\tau, N))$, $L \leftarrow U$, $L^\sharp \leftarrow 3U$
3. $(C[1], \ldots, C[m]) \xleftarrow{n} C$
4. **for** $i = 1$ **to** $\lceil m/2 \rceil - 1$ **do**
5. $\quad M[2i-1] \leftarrow E(L^\sharp \oplus C[2i-1]) \oplus C[2i]$
6. $\quad M[2i] \leftarrow E(L \oplus M[2i-1]) \oplus C[2i-1]$
7. $\quad \Sigma \leftarrow \Sigma \oplus M[2i]$
8. $\quad L \leftarrow L \oplus L^\sharp$, $L^\sharp \leftarrow 2L^\sharp$ // $L = 2^i U$, $L^\sharp = 2^i 3U$
9. **if** m **is even**
10. $\quad M[m-1] \leftarrow E(L^\sharp \oplus \underline{C[m]}) \oplus C[m-1]$
11. $\quad Z \leftarrow E(L \oplus M[m-1])$
12. $\quad M[m] \leftarrow \mathtt{msb}_{|C[m]|}(Z) \oplus C[m]$
13. $\quad \Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$
14. $\quad L^* \leftarrow L^\sharp$
15. **if** m **is odd**
16. $\quad M[m] \leftarrow \mathtt{msb}_{|C[m]|}(E(L)) \oplus C[m]$
17. $\quad \Sigma \leftarrow \Sigma \oplus \underline{M[m]}$
18. $\quad L^* \leftarrow L$
19. **if** $|C[m]| \neq n$ **then** $TE \leftarrow E(3^2 L^* \oplus \Sigma)$
20. **else** $TE \leftarrow E(7L^* \oplus \Sigma)$
21. $M \leftarrow (M[1], \ldots, M[m])$
22. **return** (M, TE)

Algorithm $\mathrm{AF}_E(A)$

1. $\Xi \leftarrow 0^n$
2. $Q \leftarrow E(0^n)$
3. $(A[1], \ldots, A[a]) \xleftarrow{n} A$
4. **for** $i = 1$ **to** $a - 1$ **do**
5. $\quad \Xi \leftarrow \Xi \oplus E(Q \oplus A[i])$
6. $\quad Q \leftarrow 2Q$
7. $\Xi \leftarrow \Xi \oplus \underline{A[a]}$
8. **if** $|A[a]| \neq n$ **then** $TA \leftarrow E(3Q \oplus \Xi)$
9. **else** $TA \leftarrow E(3^2 Q \oplus \Xi)$
10. **return** TA

Algorithm $\mathrm{Format}(\tau, N)$

1. **return** $\mathtt{n2s}(\tau \bmod n, 7) \| 0^{n-8-|N|} \| 1 \| N$

Fig. 1 Algorithms of AES-OTR with parallel ADP. Tag bit size is $0 < \tau \leq n$, and $\underline{X}$ denotes the 10^* padding of X (See Section 1.3).

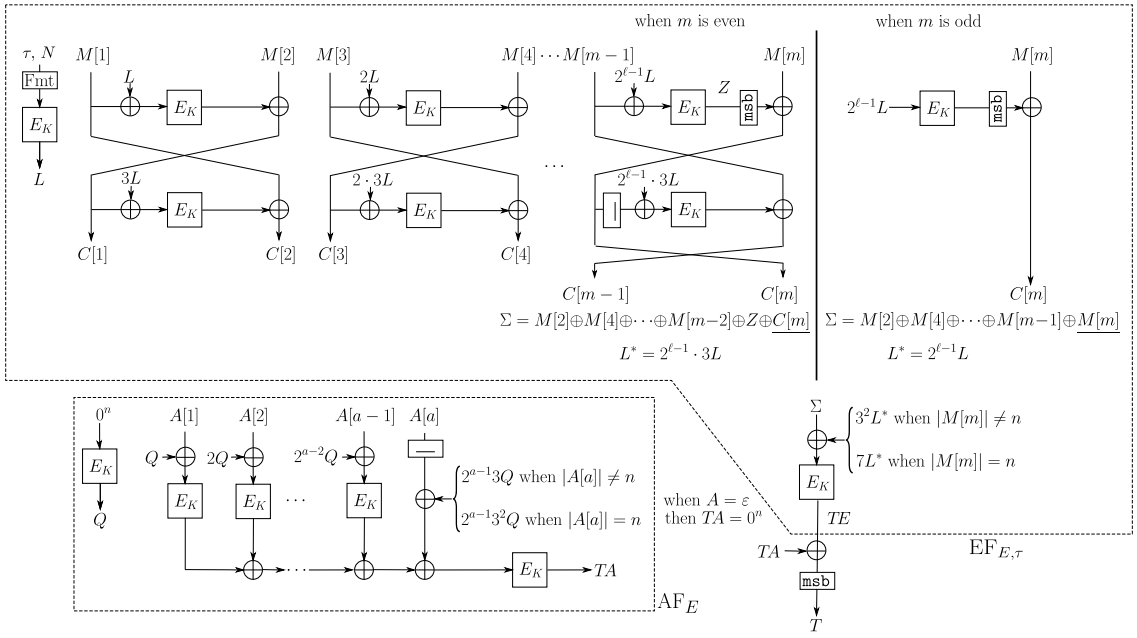

Fig. 2 Encryption of AES-OTR with parallel ADP. Fmt denotes the function Format, and a box with underline and $\underline{X}$ denote the 10^* padding of input X.

base 2. If one of the variables reaches the required amount it means a break. For Table 1, Data and Time denote $\sigma_{\texttt{priv}}$ and t' in the privacy bounds (Theorems 3.1 and Theorem 3.3). For Table 2, Data, Verify, and Time denote $\sigma_{\texttt{auth}}$, q_v, and t' in the authenticity bounds (Theorem 3.2 and Theorem 3.4). The inclusion of Verify in Table 2 is redundant since $\sigma_{\texttt{auth}}$ includes q_v and τ is 128 for the parameter sets here, however, we explicitly present it to make clear that $q_v \ll \tau$ is a security condition, which is meaningful if τ is small. Strictly speaking, the Data figures in the tables should be slightly smaller with respect to the bounds, say Data 64 in Table 1 of `aes128otrpv1` and `aes256otrpv1` is $64 - \log_2(c)$ with $c = \sqrt{6}$ from Theorem 3.1.

As an additional security goal, we claim that the length of nonce can be changed without key renewal, provided all nonces are unique for all encryptions, for both parallel and serial ADP versions. This setting does not change the security bounds of Section 3. In addition, the security of serial ADP holds as far as a pair of AD and nonce (A, N) is unique for all encryption queries, for privacy and authenticity notions.

Table 1 Security goal for confidentiality (privacy).

	`aes128otrpv1`		`aes128otrsv1`		`aes256otrpv1`		`aes256otrsv1`	
Confidentiality (Privacy)	Data 64	Time 128	Data 64	Time 128	Data 64	Time 256	Data 64	Time 256

Table 2 Security goal for integrity (authenticity).

	`aes128otrpv1`			`aes128otrsv1`			`aes256otrpv1`			`aes256otrsv1`		
Integrity (Authenticity)	Data 64	Verify 128 (τ)	Time 128	Data 64	Verify 128 (τ)	Time 128	Data 64	Verify 128 (τ)	Time 256	Data 64	Verify 128 (τ)	Time 256

3 Security Analysis

■Provable Security Paradigm. AES-OTR has a provable security based on the assumption that AES is a pseudorandom function. Below we provide a brief explanation on the security model we consider, which

Algorithm OTR-$\mathcal{E}_{E,\tau,s}(N, A, M)$

1. **if** $A \neq \varepsilon$ **then** $TA \leftarrow$ AF-S$_E(A)$
2. **else** $TA \leftarrow 0^n$
3. $(C, TE) \leftarrow$ EF-S$_{E,\tau}(N, M, TA)$
4. $T \leftarrow \mathtt{msb}_\tau(TE)$
5. **return** (C, T)

Algorithm OTR-$\mathcal{D}_{E,\tau,s}(N, A, C, T)$

1. **if** $A \neq \varepsilon$ **then** $TA \leftarrow$ AF-S$_E(A)$
2. **else** $TA \leftarrow 0^n$
3. $(M, TE) \leftarrow$ DF-S$_{E,\tau}(N, C, TA)$
4. $\widehat{T} \leftarrow \mathtt{msb}_\tau(TE)$
5. **if** $\widehat{T} = T$ **return** M
6. **else return** $\perp$

Algorithm EF-S$_{E,\tau}(N, M, TA)$

1. $\Sigma \leftarrow 0^n$
2. $U \leftarrow 2(E(\text{Format}(\tau, N)) \oplus TA)$
3. $L \leftarrow U$, $L^\sharp \leftarrow 3U$
4. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
5. **for** $i = 1$ **to** $\lceil m/2 \rceil - 1$ **do**
6. $\quad C[2i-1] \leftarrow E(L \oplus M[2i-1]) \oplus M[2i]$
7. $\quad C[2i] \leftarrow E(L^\sharp \oplus C[2i-1]) \oplus M[2i-1]$
8. $\quad \Sigma \leftarrow \Sigma \oplus M[2i]$
9. $\quad L \leftarrow L \oplus L^\sharp$, $L^\sharp \leftarrow 2L^\sharp$ // $L = 2^i U$, $L^\sharp = 2^i 3U$
10. **if** m **is even**
11. $\quad Z \leftarrow E(L \oplus M[m-1])$
12. $\quad C[m] \leftarrow \mathtt{msb}_{|M[m]|}(Z) \oplus M[m]$
13. $\quad C[m-1] \leftarrow E(L^\sharp \oplus \underline{C[m]}) \oplus M[m-1]$
14. $\quad \Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$
15. $\quad L^* \leftarrow L^\sharp$
16. **if** m **is odd**
17. $\quad C[m] \leftarrow \mathtt{msb}_{|M[m]|}(E(L)) \oplus M[m]$
18. $\quad \Sigma \leftarrow \Sigma \oplus \underline{M[m]}$
19. $\quad L^* \leftarrow L$
20. **if** $|M[m]| \neq n$ **then** $TE \leftarrow E(3^2 L^* \oplus \Sigma)$
21. **else** $TE \leftarrow E(7L^* \oplus \Sigma)$
22. $C \leftarrow (C[1], \ldots, C[m])$
23. **return** (C, TE)

Algorithm DF-S$_{E,\tau}(N, C, TA)$

1. $\Sigma \leftarrow 0^n$
2. $U \leftarrow 2(E(\text{Format}(\tau, N)) \oplus TA)$
3. $L \leftarrow U$, $L^\sharp \leftarrow 3U$
4. $(C[1], \ldots, C[m]) \xleftarrow{n} C$
5. **for** $i = 1$ **to** $\lceil m/2 \rceil - 1$ **do**
6. $\quad M[2i-1] \leftarrow E(L^\sharp \oplus C[2i-1]) \oplus C[2i]$
7. $\quad M[2i] \leftarrow E(L \oplus M[2i-1]) \oplus C[2i-1]$
8. $\quad \Sigma \leftarrow \Sigma \oplus M[2i]$
9. $\quad L \leftarrow L \oplus L^\sharp$, $L^\sharp \leftarrow 2L^\sharp$ // $L = 2^i U$, $L^\sharp = 2^i 3U$
10. **if** m **is even**
11. $\quad M[m-1] \leftarrow E(L^\sharp \oplus \underline{C[m]}) \oplus C[m-1]$
12. $\quad Z \leftarrow E(L \oplus M[m-1])$
13. $\quad M[m] \leftarrow \mathtt{msb}_{|C[m]|}(Z) \oplus C[m]$
14. $\quad \Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$
15. $\quad L^* \leftarrow L^\sharp$
16. **if** m **is odd**
17. $\quad M[m] \leftarrow \mathtt{msb}_{|C[m]|}(E(L)) \oplus C[m]$
18. $\quad \Sigma \leftarrow \Sigma \oplus \underline{M[m]}$
19. $\quad L^* \leftarrow L$
20. **if** $|C[m]| \neq n$ **then** $TE \leftarrow E(3^2 L^* \oplus \Sigma)$
21. **else** $TE \leftarrow E(7L^* \oplus \Sigma)$
22. $M \leftarrow (M[1], \ldots, M[m])$
23. **return** (M, TE)

Algorithm AF-S$_E(A)$

1. $\Xi \leftarrow 0^n$
2. $Q \leftarrow E(0^n)$
3. $(A[1], \ldots, A[a]) \xleftarrow{n} A$
4. **for** $i = 1$ **to** $a - 1$ **do**
5. $\quad \Xi \leftarrow E(A[i] \oplus \Xi)$
6. $\Xi \leftarrow \Xi \oplus \underline{A[a]}$
7. **if** $|A[a]| \neq n$ **then** $TA \leftarrow E(2Q \oplus \Xi)$
8. **else** $TA \leftarrow E(4Q \oplus \Xi)$
9. **return** TA

Algorithm Format(τ, N)

1. **return** $\mathtt{n2s}(\tau \bmod n, 7) \| 0^{n-8-|N|} \| 1 \| N$

Fig. 3 Algorithms of AES-OTR with serial ADP. Tag bit size is $0 < \tau \leq n$, and $\underline{X}$ denotes the 10^* padding of X (See Section 1.3)

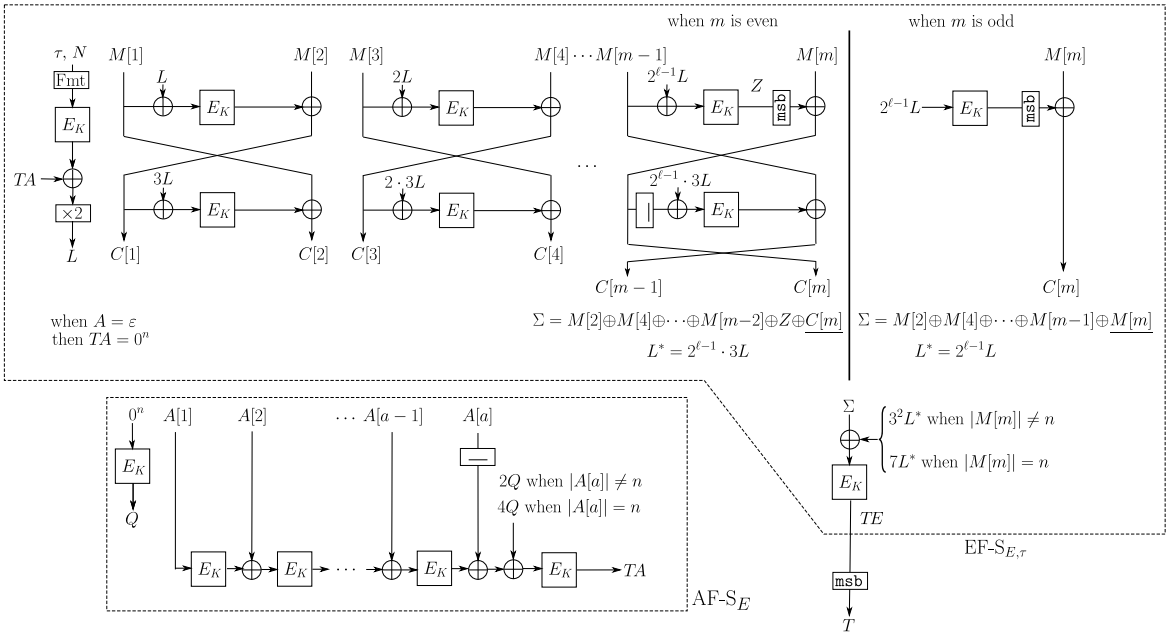

Fig. 4 Encryption of AES-OTR with serial ADP. Fmt denotes the function Format, and a box with underline and $\underline{X}$ denote the 10^* padding of input X.

is common to many nonce-based AEAD blockcipher modes.

Let $\mathsf{AE}[\tau]$ be an AEAD having τ-bit tag, where the encryption and decryption algorithms are AE-$\mathcal{E}_\tau$ and AE-$\mathcal{D}_\tau$. Then, a PRIV-adversary $\mathcal{A}$ against $\mathsf{AE}[\tau]$ accesses AE-$\mathcal{E}_\tau$, where the i-th query consists of nonce N_i, associated data A_i, and plaintext M_i. The final output of $\mathcal{A}$ is a binary variable indicating $\mathcal{A}$'s guess. We define $\mathcal{A}$'s parameter list to be (q, σ_A, σ_M), where q denotes the number of queries, and $\sigma_A \stackrel{\text{def}}{=} \sum_{i=1}^{q} |A_i|_n$ and $\sigma_M \stackrel{\text{def}}{=} \sum_{i=1}^{q} |M_i|_n$. We assume $\mathcal{A}$ is nonce-respecting, i.e., all N_is are distinct. We also define random-bit oracle, \$, which takes (N, A, M) and returns $(C, T) \xleftarrow{\$} \{0,1\}^{|M|} \times \{0,1\}^\tau$. The privacy notion for $\mathcal{A}$ is defined as

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathsf{AE}[\tau]}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{AE}\text{-}\mathcal{E}_\tau} \Rightarrow 1] - \Pr[\mathcal{A}^{\$} \Rightarrow 1]. \tag{4}$$

An AUTH-adversary $\mathcal{A}$ against $\mathsf{AE}[\tau]$ accesses AE-$\mathcal{E}_\tau$ and AE-$\mathcal{D}_\tau$, using q encryption queries and q_v decryption queries. Let $(N_1, A_1, M_1), \ldots, (N_q, A_q, M_q)$ and $(N'_1, A'_1, C'_1, T'_1), \ldots, (N'_{q_v}, A'_{q_v}, C'_{q_v}, T'_{q_v})$ be all the encryption and decryption queries made by $\mathcal{A}$. We define $\mathcal{A}$'s parameter list to be $(q, q_v, \sigma_A, \sigma_M, \sigma_{A'}, \sigma_{C'})$, where $\sigma_{A'} \stackrel{\text{def}}{=} \sum_{i=1}^{q_v} |A'_i|_n$ and $\sigma_{C'} \stackrel{\text{def}}{=} \sum_{i=1}^{q_v} |C'_i|_n$, in addition to σ_A and σ_M. The authenticity notion for the AUTH-adversary $\mathcal{A}$ is defined as

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathsf{AE}[\tau]}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{AE}\text{-}\mathcal{E}_\tau, \mathsf{AE}\text{-}\mathcal{D}_\tau} \text{ forges }], \tag{5}$$

where $\mathcal{A}$ forges if AE-$\mathcal{D}_\tau$ returns a bit string (other than $\perp$) for a decryption query (N'_i, A'_i, C'_i, T'_i) for some $1 \leq i \leq q_v$ such that $(N'_i, A'_i, C'_i, T'_i) \neq (N_j, A_j, C_j, T_j)$ for all $1 \leq j \leq q$. We assume AUTH-adversary $\mathcal{A}$ is always nonce-respecting with respect to encryption queries; using the same N for encryption and decryption queries is allowed, and the same N can be repeated within decryption queries, i.e. N_i is different from N_j for any $j \neq i$ but N'_i may be equal to N_j or $N'_{i'}$ for some j and $i' \neq i$.

The above definitions omit a parameter to specify the adversary's computation power (i.e., for unlimited computational power). If we need to specify the adversary's computation power, we will additionally define the time complexity, t, as the sum of running time of attack and the code size, under some fixed computation model. For details, see e.g. [10].

■PRP and PRF. Let $\mathrm{Func}(a,b)$ be the set of all functions : $\{0,1\}^a \to \{0,1\}^b$, and let $\mathrm{Perm}(a)$ be the set of all permutation over $\{0,1\}^a$. A uniform random function (URF) and a uniform random permutation (URP) is defined as a random function distributed uniformly random over $\mathrm{Func}(a,b)$ and $\mathrm{Perm}(a)$.

For n-bit blockcipher E with key $K \in \mathcal{K}_{ae}$, Let $\mathtt{Adv}^{\mathtt{prp}}_{E_K}(q,t)$ be the distinguishing advantage between E_K and an n-bit URP, P, using q chosen-plaintext queries and time complexity t. More formally, we have

$$\mathtt{Adv}^{\mathtt{prp}}_{E_K}(q,t) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}}\{\Pr[K \stackrel{\$}{\leftarrow} \mathcal{K}_{ae} : \mathcal{A}^{E_K} \to 1] - \Pr[\mathsf{P} \stackrel{\$}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{\mathsf{P}} \to 1]\} \tag{6}$$

where the maximum is taken for all adversaries using q chosen-plaintext queries with time complexity t. By convention we say a blockcipher E_K is a pseudorandom permutation (PRP) if $\mathtt{Adv}^{\mathtt{prp}}_{E_K}(q,t)$ is sufficiently small [23][*1]. Similarly we define

$$\mathtt{Adv}^{\mathtt{prf}}_{E_K}(q,t) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}}\{\Pr[K \stackrel{\$}{\leftarrow} \mathcal{K}_{ae} : \mathcal{A}^{E_K} \to 1] - \Pr[\mathsf{R} \stackrel{\$}{\leftarrow} \mathrm{Func}(n,n) : \mathcal{A}^{\mathsf{P}} \to 1]\} \tag{7}$$

and (conventionally) say E_K is a pseudorandom function (PRF) when $\mathtt{Adv}^{\mathtt{prf}}_{E_K}(q,t)$ is sufficiently small. Thanks to the well-known PRP-PRF switching lemma (e.g. see [13]), the both statements are exchangeable as long as $q \ll 2^{n/2}$.

■Security Bounds. First we provide the security bounds of OTR with parallel ADP.

Theorem 3.1 Fix $\tau \in \{1,\ldots,n\}$. For any PRIV-adversary $\mathcal{A}$ with parameter (q,σ_A,σ_M) and time complexity t,

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{OTR}[E_K,\tau,p]}(\mathcal{A}) \le \mathtt{Adv}^{\mathtt{prp}}_{E_K}(2\sigma_{\mathtt{priv}},t') + \frac{6\sigma^2_{\mathtt{priv}}}{2^n}$$

holds for $\sigma_{\mathtt{priv}} = q + \sigma_A + \sigma_M$ and t' is about t plus the complexity of $2\sigma_{\mathtt{priv}}$ AES encryption.

Theorem 3.2 Fix $\tau \in \{1,\ldots,n\}$. For any AUTH-adversary $\mathcal{A}$ with parameter $(q,q_v,\sigma_A,\sigma_M,\sigma_{A'},\sigma_{C'})$ and time complexity t,

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{OTR}[E_K,\tau,p]}(\mathcal{A}) \le \mathtt{Adv}^{\mathtt{prp}}_{E_K}(2\sigma_{\mathtt{auth}},t') + \frac{6\sigma^2_{\mathtt{auth}}}{2^n} + \frac{q_v}{2^\tau}$$

holds for $\sigma_{\mathtt{auth}} = q + q_v + \sigma_A + \sigma_M + \sigma_{A'} + \sigma_{C'}$ and t' is about t plus the complexity of $2\sigma_{\mathtt{auth}}$ AES encryption.

Next we provide the security bounds of OTR with serial ADP for privacy and authenticity notions.

Theorem 3.3 Fix $\tau \in \{1,\ldots,n\}$. For any CPA-adversary $\mathcal{A}$ with parameter (q,σ_A,σ_M) and time complexity t,

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{OTR}[E_K,\tau,s]}(\mathcal{A}) \le \mathtt{Adv}^{\mathtt{prp}}_{E_K}(2\sigma_{\mathtt{priv}},t') + \frac{5.5\sigma^2_{\mathtt{priv}}}{2^n}$$

holds for $\sigma_{\mathtt{priv}} = q + \sigma_A + \sigma_M$ and t' is about t plus the complexity of $2\sigma_{\mathtt{priv}}$ AES encryption.

Theorem 3.4 Fix $\tau \in \{1,\ldots,n\}$. For any CCA-adversary $\mathcal{A}$ with parameter $(q,q_v,\sigma_A,\sigma_M,\sigma_{A'},\sigma_{C'})$ and time complexity t,

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{OTR}[E_K,\tau,s]}(\mathcal{A}) \le \mathtt{Adv}^{\mathtt{prp}}_{E_K}(2\sigma_{\mathtt{auth}},t') + \frac{7.5\sigma^2_{\mathtt{auth}}}{2^n} + \frac{q_v}{2^\tau}$$

holds for $\sigma_{\mathtt{auth}} = q + q_v + \sigma_A + \sigma_M + \sigma_{A'} + \sigma_{C'}$ and t' is about t plus the complexity of $2\sigma_{\mathtt{auth}}$ AES encryption.

[*1] This is actually a convention because the formal definition of PRP is for a family of keyed permutation, and we here did not specify a formal definition of "sufficiently small".

Full proofs of all theorems appear in [25]. The difference in nonce encoding (i.e. we change it from $\underline{N}$ to Format(τ, N)) does not change the result since the proofs are maintained as long as encoded nonce and the constant 0^n are distinct, for both parallel and serial ADP versions.

4 Features

AES-OTR has the following features.

- The key is one AES key, K.
- Inverse-free, that is, encryption and decryption can be done by AES encryption function, E_K.
- Rate-1 processing for both encryption and decryption. More precisely, for a-block AD (for $a > 0$) and m-block plaintext, the number of AES calls is $a + m + 2$ (with one call can be preprocessed and cached). If $a = 0$ we need $m + 2$ calls.
- On-line, one-pass, and parallel encryption and decryption, under two-block partition. For serial ADP version, associated data processing is serial but plaintext can be processed in parallel.
- Provable security up to about $2^{n/2}$ input blocks, based on the assumption that E_K is a pseudorandom function (PRF). From the PRP-PRF switching lemma, this holds also for the assumption that E_K is a pseudorandom permutation (PRP).

■Advantages over Previous AEAD Modes. A comparison of OTR with other modes is shown in Table 3. Being rate-1 means the computation cost is halved from common rate-2 schemes. Compared with AES-GCM, AES-OTR does not require a full GF multiplier, while the computation cost of AES-OTR is comparable to the encryption mode without the blockcipher decryption, such as AES-CTR. Roughly, the computation cost for encryption is comparable to that of AES-OCB. For decryption, their performances can be different depending on the platform, see below.

In return for these attractive features, one potential drawback of OTR is that it needs two-block partition, which requires more state memories required than that of OCB, though OCB needs the additional implementation of blockcipher decryption. The parallelizability of our scheme is up to the half of the message blocks, while OCB has full parallelizability, up to the number of message blocks. On-line processing capability is restrictive as it needs buffering of consecutive two input blocks.

For memory consumption, all inverse-free modes including OTR have a similar profile, as long as the blockcipher encryption is the dominant factor. An exception is GCM since field multiplication needs large memories for fast operation. The design of OTR avoids a large memory consumption.

Table 3 A comparison of AEAD modes. Calls denotes the number of primitive calls for m-block message and a-block header and one-block nonce, without constants.

Mode	Calls	On-line	Parallel	Primitive
CCM [4]	$a + 2m$	no	no	E
GCM [6]	m [E] and $a + m$ [Mul]	yes	yes	$E, \text{Mul}^\dagger$
EAX [11]	$a + 2m$	yes	no	E
OCB [22, 29, 31]	$a + m$	yes	yes	E, E^{-1}
CCFB [24]	$a + cm$ for some $1 < c^\ddagger$	yes	no	E
OTR	$a + m$	yes¶	yes¶	E

† GF(2^n) multiplication
‡ Security degrades as c approaches 1
¶ two-block partition, serial AD for serial ADP version.

■Benefits of inverse-freeness. The features listed above are already covered by OCB mode, except that OCB needs both blockcipher encryption and decryption (more precisely the OCB decryption needs both functions). As mentioned by Iwata and Yasuda [20], inverse-freeness can contribute to efficiency and security. First, for efficiency, the initialization AES-OTR needs AES key schedule only for the forward direction. In the standard, so-called T-table implementation, we can remove some tables required for the

inverse, resulting in smaller ROM and RAM consumption on software. When side-channel protection is needed, inverse-freeness can help reduce the critical functions to protect. Moreover, AES decryption can be slower than AES encryption on some platforms (e.g., see AES implementation for Atmel AVR by Osvik et al. [28]). This property is a consequence of the initial design choice [15], focusing on inverse-free modes. The uneven performance figures of blockcipher enc/dec functions is undesirable in practice, when the mode uses both functions.

This phenomena, however, is not always true. Most notably, recent Intel's CPUs have AES instructions called AESNI, which enables the identical performance for AES encryption and decryption, and AMD CPUs have equivalent ones, too. Therefore, when our proposal uses AESNI, the performance would be roughly similar to that of AES-OCB with AESNI, for both encryption and decryption, though the complexity outside the blockcipher may degrade the result. We have other SW platforms where hardware AES is available but decryption is slower (e.g., [18]), or only encryption function is available (e.g., [16]). Basically, the value of our proposal is *not* to provide the fastest operation on modern CPUs, instead, to increase the availability of rate-1 performance for various platforms, using single algorithm based on the minimal cryptographic primitive and assumption.

For hardware the algorithm is parallelizable and also pipelinable. AES-OTR's inverse-freeness significantly contributes in hardware size reduction for various forms of hardware implementation from standard round-based implementation to extremely high speed implementations using pipeline or multiple AES cores.

Next, inverse-freeness can contribute to security. Usually the security of a mode using both enc/dec functions of a blockcipher, denoted by E and E^{-1}, needs (E, E^{-1}) to be a strong pseudorandom permutation (Strong PRP or SPRP). This holds true for the original security proofs of all versions of OCB [22,29,31], though a recent work of Aoki and Yasuda [9] showed a relaxation on the security condition for the blockcipher used by OCB. In contrast, when the mode uses only E, the security assumption is relaxed to PRP or PRF.

■Security. AES-OTR has a provable security reduction to the pseudorandomness of AES, which is a quite popular assumption used by many cryptographic designs. The proved security bounds show that one can not break the scheme in the senses of privacy and authenticity, up to around the birthday bound, i.e. 2^{64} blocks. Here, however, we have to warn that the security is proved for the standard nonce-respecting adversary, i.e. the encryption never processes duplicate nonces. We do not claim any security guarantee for adversaries beyond this condition, except a slight relaxation for serial ADP, shown in Section 2, which says that the security is preserved as long as a pair (A, N) is unique, i.e., it can work as nonce.

■Justifications for Recommended Parameter Sets. The nonce length of 12 bytes is a reasonable choice for simplicity, and is widely employed by many protocols. The tag length of 16 bytes offers high-security against even intensive forgery attempts. The AES key length of 128 bits provides adequate security against key exhaustive searches, and 256 key bits are sometimes required for extra security. For parallel ADP we need more memory than serial ADP, hence the latter is preferable for serial computing environment. In addition the computation of serial ADP is simpler than parallel ADP. See Section 5.

In addition, we specify the minimum tag length to 4 bytes, though the recommended parameter sets have 16-byte tag. Such short tag is usually dangerous, however, still useful to supress the communication overhead with an appropriate limitation on the amount of data processed for one key. In fact we can find many examples of 4-byte tags in the realm of low-power wireless sensor networks, e.g., 6LoWPAN and Zigbee, both using AES-CCM with tag length between 4 to 16 bytes [2,3].

5 Design Rationale

Our goal is to provide a fast and compact AEAD suitable for various platforms, using a minimum cryptographic primitive, i.e., encryption function of a blockcipher. We choose AES as the underlying blockcipher, because the security of AES has been extensively studied. The cipher achieves rate-1 parallelizable processing without using AES inverse. The computation cost is comparable to encryption

modes using AES encryption function (e.g. CTR). Since we can not achieve privacy notion of AE with fewer AES calls than AES-CTR, this implies that the computation cost of AES-OTR is asymptotically minimum as a mode of AES encryption function.

■Masks. For masking applied to AES inputs inside AES-OTR, we employed constant GF multiplications, called GF doublings. This is used by many schemes. For hardware, doubling is simple. For software, doubling is basically not simple as it requires bit-shift of whole 128-bit block. However, recent studies reported that the optimized doubling software implementation can be fast [8]. Considering this we employ on-the-fly doubling as a practical masking option. We remark that using the same mask for the round functions in a two-round Feistel does not work. This is because the two-round Feistel then becomes an involution, which allows the adversary to control the checksum value in the decryption, hence breaks authenticity. We also remark that the all masks for EF_E depend on N, hence do not allow precomputation before N is given, which is different from the latest OCB3 [22]. The reason of our choice is that we want our scheme not to waste $E_K(0^n)$ when AD is always empty, and not to require a large RAM for precomputation, considering applicability to constrained devices. Our goal is not to ultimately focus on one platform, hence we consider keeping the amount of precomputation small is good to achieve a balanced performance under multiple platforms.

OTR's blockcipher masks can be generated by various ways. We suggest some here.

- A basic option is to follow Figs. 1 and 3, which needs one doubling per two message blocks, using two state blocks.
- If we want to generate masks in a parallel computation, the simplest way is to take doublings independently for the first and second rounds, however a faster method is possible as shown by [14] since the second-round masks for encryption can be generated by taking XORs of the first-round masks. That is, the second-round masks, $(3L, 2\cdot 3L, 2^2 3L, \dots,)$, are generated from the first-round masks $(L, 2L, 2^2 L, \dots,)$ as $L \oplus 2L \to 3L$, $2^2 L \oplus 2L \to 2 \cdot 3L$, and so on. Decryption can be done similarly. This method is useful for hardware or SIMD batch doublings on X86 CPUs [7, 27].
- If we want to reduce memory consumption, we can compute masks completely serial, using one state block, with slight more complex, hardware-friendly bit operations. For example we need routines for $X \to 3X$ and $3X \to 2X$ for encryption.

■Associated Data Processing. We have two versions depending on associated data processing, ADP. For parallel ADP version, the processing of AD is based on (a variant of) PMAC, and the computation can be done in parallel to the processing of plaintext/ciphertext. This is to maximize the parallel computation capability. We can efficiently handle static AD, and employ counter-nonce caching due to Wu (mentioned by Bernstein [12]), if needed. For serial ADP version, the processing of AD is based on CMAC, hence is inherently serial. At the cost of losing parallelizability for AD, this version reduces the computation cost of ADP in serial environment. Also, the point where CMAC output (TA) is xored is chosen to reduce the required state memory, while keeping the capability of static AD and counter-nonce caching. These design choices make AES-OTR with serial ADP version suitable to, e.g., embedded software.

The designer has not hidden any weaknesses in this cipher.

6 Intellectual Property

We have a pending patent application relating to our AES-OTR proposal, JP application No. 2013-161446. In case that AES-OTR is included into the final portfolio, we are willing to provide to implementors, solely for the purpose of implementing AES-OTR, a royalty-free non-exclusive license under the patents issuing on such patent application to the extent such patents are essential to implement AES-OTR as set forth in the final portfolio, provided said implementor extends a reciprocal royalty-free license. 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.

Changes

AES-OTR v1 (2014.3.15) : Initial submission.

AES-OTR v1.1 (2015.1.19) : Added clarification on doubling in Section 1.3 and updates on Section 6 (Intellectual Property) and References.

AES-OTR v2 (2015.8.29) : A change in nonce processing (encryption). It includes an encode of tag length information in the same manner to OCB. Figures 1 to 4 are updated with respect to this change. Add some supporting texts at Section 1.3, Section 2, and Section 4 with additional references. Minor cover edit and fixed typos.

AES-OTR v3 (2016.4.4) : Bost and Sanders [14] pointed out that the mask generations of the previous versions are not following Rogaway's XE mode [29], which leads to a flaw in the security proof. The designer appreciates their work. Reflecting it, the masking constants are changed to follow XE. Corresponding figures and pseudo-codes are also updated.

AES-OTR v3.1 (2016.9.1) : Specified the targeted use case in Section 1.2. No algorithmic changes.

References

[1] Crypto Forum Research Group Discussion Archive, `http://www.ietf.org/mail-archive/web/cfrg/current/msg03433.html/`

[2] IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals, `https://tools.ietf.org/html/rfc4919/`

[3] ZigBee Alliance, `http://www.zigbee.org/`

[4] Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. National Institute of Standards and Technology (NIST) Special Publication 800-38C (2004)

[5] Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. National Institute of Standards and Technology (NIST) Special Publication 800-38B (2005)

[6] Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. National Institute of Standards and Technology (NIST) Special Publication 800-38D (2007)

[7] Aoki, K.: Optimization of mode implementations on Sandy Bridge. Symposium on Cryptography and Information Security (SCIS) 2013, in Japanese

[8] Aoki, K., Iwata, T., Yasuda, K.: How Fast Can a Two-Pass Mode Go? A Parallel Deterministic Authenticated Encryption Mode for AES-NI. DIAC 2012: Directions in Authenticated Ciphers (2012), available from `http://hyperelliptic.org/DIAC/`

[9] Aoki, K., Yasuda, K.: The Security of the OCB Mode of Operation without the SPRP Assumption. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec. Lecture Notes in Computer Science, vol. 8209, pp. 202–220. Springer (2013)
[10] Bellare, M., Kilian, J., Rogaway, P.: The Security of the Cipher Block Chaining Message Authentication Code. J. Comput. Syst. Sci. 61(3), 362–399 (2000), http://dx.doi.org/10.1006/jcss.1999.1694
[11] Bellare, M., Rogaway, P., Wagner, D.: The EAX Mode of Operation. In: Roy and Meier [33], pp. 389–407
[12] Bernstein, D.J., Schwabe, P.: New AES Software Speed Records. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 5365, pp. 322–336. Springer (2008)
[13] Black, J., Rogaway, P.: CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. In: Bellare, M. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1880, pp. 197–215. Springer (2000)
[14] Bost, R., Sanders, O.: Trick or Tweak: On the (In)security of OTR's Tweaks. Cryptology ePrint Archive, Report 2016/234 (2016)
[15] Daemen, J., Rijmen, V.: AES Proposal: Rijndael (1999)
[16] Didla, S., Ault, A., Bagchi, S.: Optimizing AES for embedded devices and wireless sensor networks. In: de Leon, M.P. (ed.) TRIDENTCOM. p. 4. ICST (2008)
[17] Eichlseder, M.: Remark on variable tag lengths and OMD (2014/04/26), Cryptographic Competitions – Google Groups, https://groups.google.com/forum/#!forum/crypto-competitions/
[18] Gouvêa, C.P.L., López, J.: High Speed Implementation of Authenticated Encryption for the MSP430X Microcontroller. In: Hevia, A., Neven, G. (eds.) LATINCRYPT. Lecture Notes in Computer Science, vol. 7533, pp. 288–304. Springer (2012)
[19] Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC. In: Johansson, T. (ed.) FSE. Lecture Notes in Computer Science, vol. 2887, pp. 129–153. Springer (2003)
[20] Iwata, T., Yasuda, K.: BTM: A Single-Key, Inverse-Cipher-Free Mode for Deterministic Authenticated Encryption. In: Jr., M.J.J., Rijmen, V., Safavi-Naini, R. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 5867, pp. 313–330. Springer (2009)
[21] Krovetz, T., Rogaway, P.: CAESAR submission OCB (v1), http://competitions.cr.yp.to/round1/ocbv1.pdf/
[22] Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In: Joux, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 6733, pp. 306–327. Springer (2011)
[23] Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM J. Comput. 17(2), 373–386 (1988)
[24] Lucks, S.: Two-Pass Authenticated Encryption Faster Than Generic Composition. In: Gilbert, H., Handschuh, H. (eds.) FSE. Lecture Notes in Computer Science, vol. 3557, pp. 284–298. Springer (2005)
[25] Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions (full version). IACR Cryptology ePrint Archive 2013, 628 (2013), http://eprint.iacr.org/2013/628
[26] Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology - EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 275–292. Springer (2014)
[27] Minematsu, K., Shigeri, M., Kubo, H.: AES-OTR v2. Directions in Authenticated Ciphers (DIAC) 2015
[28] Osvik, D.A., Bos, J.W., Stefan, D., Canright, D.: Fast Software AES Encryption. In: Hong, S., Iwata, T. (eds.) FSE. Lecture Notes in Computer Science, vol. 6147, pp. 75–93. Springer (2010)
[29] Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 3329, pp. 16–31. Springer (2004)
[30] Rogaway, P.: Nonce-Based Symmetric Encryption. In: Roy and Meier [33], pp. 348–359
[31] Rogaway, P., Bellare, M., Black, J.: OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. 6(3), 365–403 (2003)

[32] Rogaway, P., Wagner, D.: A Critique of CCM. IACR Cryptology ePrint Archive 2003, 70 (2003), http://eprint.iacr.org/2003/070
[33] Roy, B.K., Meier, W. (eds.): Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers, Lecture Notes in Computer Science, vol. 3017. Springer (2004)