Tweaking Generic OTR to Avoid Forgery Attacks

Hassan Qahur Al Mahri, Leonie Simpson, Harry Bartlett, Ed Dawson, and Kenneth Koon-Ho Wong

Queensland University of Technology, George St, Brisbane 4000, Australia
hassan.mahri@hdr.qut.edu.au
{lr.simpson,h.bartlett,e.dawson,kk.wong}@qut.edu.au

Abstract. This paper considers the security of the Offset Two-Round (OTR) authenticated encryption mode [9] with respect to forgery attacks. The current version of OTR gives a security proof for specific choices of the block size (n) and the primitive polynomial used to construct the finite field $\mathbb{F}_{2^n}$. Although the OTR construction is generic, the security proof is not. For every choice of finite field the distinctness of masking coefficients must be verified to ensure security. In this paper, we show that some primitive polynomials result in collisions among the masking coefficients used in the current instantiation, from which forgeries can be constructed. We propose a new way to instantiate OTR so that the masking coefficients are distinct in every finite field $\mathbb{F}_{2^n}$, thus generalising OTR without reducing the security of OTR.

Keywords: Authenticated encryption, OTR, confidentiality, integrity, forgery attack, tweakable block cipher, symmetric encryption, AEAD

1 Introduction

Block ciphers are fundamental cryptographic primitives used in many encryption algorithms and message authentication codes. A conventional block cipher encryption algorithm E accepts two inputs: a secret and random k-bit key K and an n-bit input string M. The output is an n-bit string C, so the block cipher is represented as a map $E : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ where $\mathcal{K}$ is the key space.

To process large amounts of data, the message is divided into blocks of size n, and a block cipher has to be used in an appropriate mode of operation. Various block cipher modes of operation have been suggested, either to provide confidentiality or integrity assurance, or to provide both confidentiality and integrity assurance in a single design in a notion called Authenticated Encryption (AE). In Authenticated Encryption with Associated Data (AEAD), some portions of the message (the associated data) do not require confidentiality but still require integrity assurance.

TWEAKABLE BLOCK CIPHERS. Liskov, Rivest and Wagner [7] introduced another notion of block ciphers called tweakable block ciphers $\widehat{E}$. These ciphers

take three inputs: key K, input message M and tweak T. Tweakable block ciphers can be represented as a map $\widehat{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \rightarrow \{0,1\}^n$ where $\mathcal{T}$ is the tweak space. The purpose of the tweak is to differentiate messages and it should be easier to change the tweak rather than changing the key. Liskov, Rivest and Wagner suggest two approaches for constructing tweakable block ciphers that are provably secure as long as the underlying block cipher is secure. These constructions can be represented as $E(\Delta_h \oplus E(M_i))$ and $E(M_i \oplus \Delta_h) \oplus \Delta_h$ where Δ_h is a universal hash function operating as the tweak. However, these constructions need two keys that should be independent of each other.

DOUBLING MASKING TECHNIQUE. Halevi and Rogaway [5] proposed a tweakable block cipher mode of operation called EME that provides only confidentiality. This mode uses the doubling masking technique, where a secret masking value is used in processing each block. This secret value is initially obtained as $L = 2E(0^n)$, and then each time a different value is needed the previous value of L is doubled. This results in a series of masking values: $2L, 2^2L, 2^3L, \ldots, 2^mL$. The multiplication is performed in the finite field $\mathbb{F}_{2^n}$ by multiplying two input polynomials and finding the reminder modulo a primitive polynomial.

The doubling masking technique is very fast and efficient. In hardware implementation, the doubling is equivalent to a conditional of either a shift or shift and XOR operations. When $n = 128$ and the finite field $\mathbb{F}_{2^{128}}$ is constructed using the commonly used primitive polynomial $f(x) = x^{128} + x^7 + x^2 + x + 1$, the doubling is as follows:

$$2L = \begin{cases} L \ll 1 & \text{if MSB}(L) = 0 \\ (L \ll 1) \oplus 0^{120}10000111 & \text{if MSB}(L) = 1 \end{cases}$$

where $\ll$ is a 1-bit logical left shift operation and MSB stands for the most significant bit.

XE AND XEX CIPHERS. Using the sequence of masking values $2L, 2^2L, \ldots, 2^mL$, Rogaway [11] describes two new approaches for tweakable block ciphers. These are known as XE and XEX ciphers, and are represented as $\widehat{E} : E(M_i \oplus 2^{i-1}L)$ or $\widehat{E} : E(M_i \oplus 2^{i-1}L) \oplus 2^{i-1}L$ respectively. Rogaway proves that these designs are secure up to the birthday bound for a certain range of i values. These designs use a single key for both the block encryption operation and to initialise the sequence of masking values used as the tweaks.

When a new value is needed which is outside the range of masking values $2L$, 2^2L,..., 2^mL, a value $2^{huge}L$ is used such that *huge* is much greater than m. For the primitive polynomial $f(x) = x^{128} + x^7 + x^2 + x + 1$, Rogaway chooses 2^{huge} as 3 and shows that this is far away from the offsets $2L, 2^2L, \ldots, 2^mL$. In addition, 3 is easy to calculate as $3 = 2 \oplus 1$; therefore, $3L$ can be XOR-ed with the checksum of plaintext blocks to obtain the authentication tag as in OCB1 [11].

Note however that 3 might not be equivalent to 2^{huge} when a different primitive polynomial is used. Collisions between the masks $3L$ and 2^jL can be found

in such cases and lead to simple forgery attacks. Because of this, the choice of values for 2^{huge} must be investigated for every choice of finite field and its distinctness from the series of masking values must be verified.

OTR MODE. Several block cipher modes of operation that have been proposed for AEAD use the doubling masking technique as in XE and XEX ciphers. One such mode is Offset Two-Round (OTR) [9] proposed by Minematsu and defined for any block size n. A version of OTR mode called AES-OTR [8] was submitted to the CAESAR competition [2]. The security proof of OTR requires that all input masks are distinct; however, the masks used in OTR and AES-OTR have only been proved to be distinct for a specific choice of n and the primitive polynomial defining the finite field.

OUR CONTRIBUTION. Firstly, we show that the current instantiation of OTR uses masking coefficients that are not always distinct in fields based on other primitive polynomials, including when $n \neq 128$. We show that using the current instantiation with other primitive polynomials can result in non-distinct masking values that can be exploited in forgery attacks against the scheme. This is a problem with most modes that use the doubling masking technique.

Secondly, we propose an alternative set of masking coefficients so that OTR can use the same set of coefficients for any block size n and any primitive polynomial, without affecting the security provided by this scheme. That is, our work generalises the OTR mode using the technique of doubling masking, and removes the requirement for the user to perform huge prior calculations in order to ensure that the masks do not overlap.

Note that this work does not imply that OTR mode or AES-OTR are insecure. Note that this solution may also apply to other similar block cipher modes that use the doubling masking technique, such as OCB1 [11], ELmD [4] and AES-COPA [1].

2 Basic Notations

For simplicity and consistency, we follow the notation used in the original OTR document [10].

$\{0,1\}^*$	: the set of all finite-length binary strings.
ε	: the empty string.
K	: k-bit key used for the block cipher and tweak initialisation.
n	: the block length of the block cipher.
N	: the nonce that is changed for each message.
m	: the number of blocks in the plaintext message.
l	: the number of chunks of two blocks in the plaintext message.
$M[2i-1]$	: the odd block in the i^{th} chunk of the plaintext message.
$M[2i]$	: the even block in the i^{th} chunk of the plaintext message.
$C[2i-1]$	: the odd block in the i^{th} chunk of the corresponding ciphertext message.
$C[2i]$	: the even block in the i^{th} chunk of the corresponding ciphertext message.
A	: the associated data that need only authentication.
$\|X\|$	: the length of the string X in bits.

$X||Y$: the concatenation of the strings X and Y.
$|X|_a$: $\max\{\lceil |X|/a \rceil, 1\}$.
$\overleftarrow{}^n X$: returns $(X[1], X[2], \ldots, X[x])$ where $x = |X|_n$, $|X[i]| = n$ for $i < x$ and $|X[x]| \leq n$.
$\underline{X}$: the 10^* padding written as $X \| 10^{n-|X|-1}$.
$\mathbf{msb}_c(X)$: the first c bits of X provided that $|X| \geq c$.
E : the block cipher encryption function under the key K.
TA : the authentication tag obtained from associated data.
TE : the authentication tag obtained from plaintext message.
T : the τ-bit final authentication tag of OTR scheme.

3 OTR Description

Offset Two-round (OTR) is an authenticated encryption block cipher mode that is online, one-pass and each segment of two consecutive blocks can be processed in parallel. OTR mode has a similar structure to OCB mode [11], but OTR uses only the forward function of the block cipher for both encryption and decryption algorithms. The OTR operation is illustrated in Table 1 and Fig. 1.

The OTR algorithm accepts the following inputs: Key $K \in \{0,1\}^k$, Nonce $N \in \{0,1\}^j$ for $1 \leq j \leq n-1$, Associated Data $A \in \{0,1\}^*$ and Plaintext $M \in \{0,1\}^*$ and has the following outputs: Ciphertext $C \in \{0,1\}^*$ and Tag $T \in \{0,1\}^\tau$. The OTR encryption algorithm consists of two algorithms known as cores: an encryption core EF_E and authentication core AF_E. The OTR encryption core divides a plaintext message M into chunks, each containing two plaintext blocks. Then, each chunk is encrypted using two different masks. These two masks are doubled to obtain other two masks for the next chunk and so on.

The authentication core can process the associated data in either of the two ways: parallel or serial. OTR uses a variant of the PMAC1 scheme [11] to authenticate associated data in parallel, and uses a variant of the OMAC mode [6] to authenticate associated data serially.

The authentication tag T in OTR is generated in two different ways. For parallel associated data, a dedicated mask (depending on the last chunk) is XOR-ed with the checksum of plaintext blocks and the result is encrypted to obtain TE. The final tag T is obtained by XOR-ing TE with the resultant tag TA of authenticating the associated data. ($T = TE \oplus TA$.) For serial associated data, the associated data tag TA is used with the Nonce to obtain the secret tweak L. In this case, the plaintext tag TE will be the final tag T.

4 The Current Instantiation of OTR Mode

Initially, OTR was designed using different instantiation values for the masking coefficients. A proof that OTR is secure for the instantiation when $n = 128$ and using the primitive polynomial $f(x) = x^{128} + x^7 + x^2 + x + 1$ is given in [10].

As a general scheme, OTR is designed to work with any block size n and any finite field $\mathbb{F}_{2^n}$. However, to obtain security assurance for a different finite

Table 1. OTR algorithm [9].

Algorithm 1: OTR Encryption Core	**Algorithm 2**: OTR Decryption Core
1. $\Sigma \leftarrow 0^n$	1. $\Sigma \leftarrow 0^n$
2. $L \leftarrow E(\underline{N})$	2. $L \leftarrow E(\underline{N})$
3. $(M[1], \ldots, M[m]) \xleftarrow{n} M,\ l = \lceil m/2 \rceil$	3. $(C[1], \ldots, C[m]) \xleftarrow{n} C,\ l = \lceil m/2 \rceil$
4. **for** $i = 1$ **to** $l - 1$ **do**	4. **for** $i = 1$ **to** $l - 1$ **do**
5. $C[2i-1] \leftarrow E(2^{i-1}L \oplus M[2i-1]) \oplus M[2i]$	5. $M[2i-1] \leftarrow E(2^{i-1}3L \oplus C[2i-1]) \oplus C[2i]$
6. $C[2i] \leftarrow E(2^{i-1}3L \oplus C[2i-1]) \oplus M[2i-1]$	6. $M[2i] \leftarrow E(2^{i-1}L \oplus M[2i-1]) \oplus C[2i-1]$
7. $\Sigma \leftarrow \Sigma \oplus M[2i]$	7. $\Sigma \leftarrow \Sigma \oplus M[2i]$
8. **if** m **is even**	8. **if** m **is even**
9. $Z \leftarrow E(2^{l-1}L \oplus M[m-1])$	9. $M[2m-1] \leftarrow E(2^{l-1}3L \oplus \underline{C[m]}) \oplus C[m-1]$
10. $C[m] \leftarrow \mathrm{msb}_{\lvert M[m]\rvert}(Z) \oplus M[m]$	10. $Z \leftarrow E(2^{l-1}L \oplus M[m-1])$
11. $C[2m-1] \leftarrow E(2^{l-1}3L \oplus \underline{C[m]}) \oplus M[m-1]$	11. $M[m] \leftarrow \mathrm{msb}_{\lvert M[m]\rvert}(Z) \oplus C[m]$
12. $\Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$	12. $\Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$
13. **if** m **is odd**	13. **if** m **is odd**
14. $C[m] \leftarrow \mathrm{msb}_{\lvert M[m]\rvert}(E(2^{l-1}L)) \oplus M[m]$	14. $M[m] \leftarrow \mathrm{msb}_{\lvert M[m]\rvert}(E(2^{l-1}L)) \oplus C[m]$
15. $\Sigma \leftarrow \Sigma \oplus \underline{M[m]}$	15. $\Sigma \leftarrow \Sigma \oplus \underline{M[m]}$
16. **if** m **is even and** $\lvert M[m]\rvert \neq n$	16. **if** m **is even and** $\lvert M[m]\rvert \neq n$
17. $TE \leftarrow E(2^{l-1}3^3L \oplus \Sigma)$	17. $TE \leftarrow E(2^{l-1}3^3L \oplus \Sigma)$
18. **if** m **is even and** $\lvert M[m]\rvert = n$	18. **if** m **is even and** $\lvert M[m]\rvert = n$
19. $TE \leftarrow E(7.3.2^{l-1}L \oplus \Sigma)$	19. $TE \leftarrow E(7.3.2^{l-1}L \oplus \Sigma)$
20. **if** m **is odd and** $\lvert M[m]\rvert \neq n$	20. **if** m **is odd and** $\lvert M[m]\rvert \neq n$
21. $TE \leftarrow E(2^{l-1}3^2L \oplus \Sigma)$	21. $TE \leftarrow E(2^{l-1}3^2L \oplus \Sigma)$
22. **if** m **is odd and** $\lvert M[m]\rvert = n$	22. **if** m **is odd and** $\lvert M[m]\rvert = n$
23. $TE \leftarrow E(7.2^{l-1}L \oplus \Sigma)$	23. $TE \leftarrow E(7.2^{l-1}L \oplus \Sigma)$
24. $C \leftarrow (C[1], \ldots, C[m])$	24. $M \leftarrow (M[1], \ldots, M[m])$
25. **return** (C, TE)	25. **return** (M, TE)
Algorithm 3: OTR Authentication with Parallel A	**Algorithm 4**: OTR Authentication with Serial A
1. $\Xi \leftarrow 0^n$	1. $\Xi \leftarrow 0^n$
2. $Q \leftarrow E(0)$	2. $Q \leftarrow E(0)$
3. $(A[1], \ldots, A[a]) \xleftarrow{n} A$	3. $(A[1], \ldots, A[a]) \xleftarrow{n} A$
4. **for** $i = 1$ **to** $a - 1$ **do**	4. **for** $i = 1$ **to** $l - 1$ **do**
5. $\Xi \leftarrow \Xi \oplus E(Q \oplus A[i])$	5. $\Xi \leftarrow E(\Xi \oplus A[i])$
6. $Q \leftarrow 2Q$	
7. $\Xi \leftarrow \Xi \oplus \underline{A[a]}$	6. $\Xi \leftarrow \Xi \oplus \underline{A[a]}$
8. **if** $\lvert M[m]\rvert \overline{\neq n}$ **then** $TA \leftarrow E(3Q \oplus \Xi)$	7. **if** $\lvert M[m]\rvert \overline{\neq n}$ **then** $TA \leftarrow E(2Q \oplus \Xi)$
9. **else** $TA \leftarrow E(3^2Q \oplus \Xi)$	8. **else** $TA \leftarrow E(4Q \oplus \Xi)$
10. **return** TA	9. **return** TA

field, the user has to prove that the chosen masks for that instantiation of OTR are distinct and do not overlap. This requires discrete log computations. Using discrete log computation, Rogaway proves in [11] that certain sets of masks are distinct from each other and provide unique representation. He considers very specific choices: when $n = 128$ or $n = 64$ and when the finite field is based on certain commonly used primitive polynomials.

Bost and Sanders [3] showed trivial collisions between the OTR input masks can be found when special forms of primitive polynomial are used. These collisions can be exploited in practical forgery attacks. They suggested the use of different masking coefficients chosen from the set given by Rogaway in [11]. Accordingly, Minematsu updated the OTR instantiation coefficients [9] and noted that care must be taken in specifying the masking coefficients for other choices of n and of primitive polynomials defining $\mathbb{F}_{2^n}$.

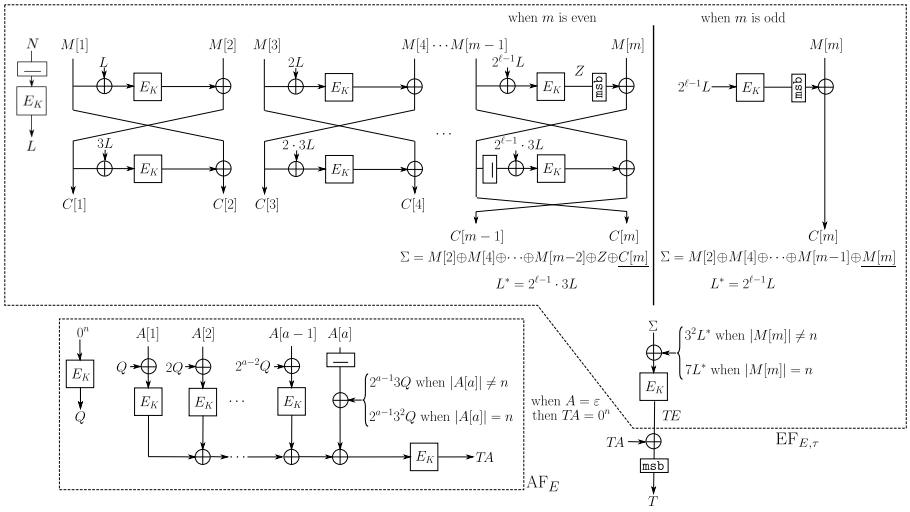

Fig. 1. OTR encryption operation with parallel associated data [9].

For generic instantiations of OTR using block sizes and primitive polynomials other than those already examined, there is a risk that a user may not select suitable masking coefficients for the instantiation. This open problem motivates us to seek a more robust definition of OTR in which the masking coefficients are distinct for any choice of finite field. Note that this will be applicable to OTR and also to any design which uses the doubling masking technique.

In our analysis, we take a similar approach to Bost and Sanders' work [3]. We consider two special forms of primitive polynomial, different to those discussed in [3], and which lead to a collision for the currently used masking coefficients.

Case 1: Primitive polynomial of the form $f(x) = x^n + x + 1$.

Many primitive polynomials can be found in this trinomial form $f(x) = x^n + x + 1$ [12]. In this case, x^n will be equal to $x+1$. That is, 3 will be equivalent to 2^n and it is not 2^{huge} as the current instantiation assumes. Therefore, a collision can be found between $3L$ and 2^nL as long as $l > n$.

Case 2: Primitive polynomial of the form $f(x) = x^n + x^2 + 1$.

This is another form of trinomial, and in this case, x^n will be equal to $x^2 + 1$. That is, 5 (which is equal to 3^2 in $\mathbb{F}_{2^n}$) will be equivalent to 2^n. In OTR, 3^2 is used when the last block $M[m]$ is not a full block ($|M[m]| \neq n$).

5 Proposed Attacks

In this section, we show how collisions between the masks can be exploited to breach the integrity assurance of OTR. We consider the two cases from Sect. 4 separately. Our analysis assumes a man-in-the-middle attack model where the attacker is able to intercept and alter messages before sending them on to the in-

tended recipient. We assume that the attacker knows both the plaintext message and its corresponding ciphertext using a single query of the OTR oracle.

Suppose that a plaintext message M is as follows:

$$M = (M[1], M[2], \ldots, M[m])$$

such that the number of blocks $m = |M|_n$ is odd and the number of chunks of two blocks $l > n + 1$ where $l = \lceil m/2 \rceil$. This message is encrypted using OTR and results in the ciphertext C:

$$C = (C[1], C[2] \ldots, C[m]).$$

5.1 Case 1 collisions

In this case, as noted in Sect. 4, $3L = 2^n L$. Suppose that the last block is a full block ($|M[m]| = n$). A forged ciphertext message $C^\star$ can be constructed as follows:

$$\begin{aligned}
C^\star[1] &= M[2(n+1)-1] \\
C^\star[2] &= M[1] \oplus M[2(n+1)] \oplus C[2(n+1)-1] \\
&= M[1] \oplus E(M[2(n+1)-1] \oplus 2^n L) \\
C^\star[i] &= C[i], \quad 3 \leq i < m \\
C^\star[m] &= C[m] \oplus C[1] \oplus M[2(n+1)-1]
\end{aligned}$$

Decrypting $C^\star$ will give the same value for all plaintext blocks except for $M^\star[2]$ and $M^\star[m]$ as follows:

$$\begin{aligned}
M^\star[1] &= E(C^\star[1] \oplus 3L) \oplus C^\star[2] \\
&= E(M[2(n+1)-1] \oplus 3L) \oplus M[1] \oplus E(M[2(n+1)-1] \oplus 2^n L) \\
&= M[1] \\
M^\star[2] &= E(M^\star[1] \oplus L) \oplus C^\star[1] = E(M[1] \oplus L) \oplus M[2(n+1)-1] \\
M^\star[m] &= E(2^{l-1} L) \oplus C^\star[m] \\
&= E(2^{l-1} L) \oplus C[m] \oplus C[1] \oplus M[2(n+1)-1] \\
&= M[m] \oplus C[1] \oplus M[2(n+1)-1]
\end{aligned}$$

Let Σ' be the checksum of all even plaintext blocks of message M except $M[2]$ and the last block $M[m]$. That is,

$$\Sigma' = \Sigma \oplus M[2] \oplus M[m]$$

Therefore, the checksum of plaintext blocks for the forged message is:

$$\begin{aligned}
\Sigma^\star &= \Sigma' \oplus M^\star[2] \oplus M^\star[m] \\
&= \Sigma' \oplus E(M[1] \oplus L) \oplus M[2(n+1)-1] \oplus M[m] \oplus C[1] \oplus M[2(n+1)-1] \\
&= \Sigma' \oplus E(M[1] \oplus L) \oplus C[1] \oplus M[m] \\
&= \Sigma' \oplus M[2] \oplus M[m] \\
&= \Sigma
\end{aligned}$$

Both C and the forged message $C^\star$ produce the same checksum value. Thus, $C^\star$ will produce the same tag T as C, and will be accepted as genuine.

5.2 Case 2 collisions

For this case, suppose that the message M has also the following features: $|M[m]| \neq n$, $M[1] = 0^n$ and $M[2(n+1)-1] = (\{1,0\}^j \parallel 10^*)$ for $1 \leq j < n$. If M is encrypted using OTR the pair (C, TE) is obtained. From this pair, an attacker can calculate $E(L) = C[1] \oplus M[2]$. A new pair $(C^\star, TE^\star)$ can be constructed from this such that $C^\star = C^\star[1] = \mathbf{msb}_j(M[2(n+1)-1] \oplus E(L))$ and the tag $TE^\star = C[2(n+1)-1] \oplus M[2(n+1)]$.

Decrypting the forged pair $(C^\star, TE^\star)$ will give:

$$\begin{aligned} M^\star[1] &= \mathbf{msb}_j(E(L)) \oplus C^\star[1] \\ &= \mathbf{msb}_j(E(L)) \oplus \mathbf{msb}_j(M[2(n+1)-1] \oplus E(L)) \\ &= \mathbf{msb}_j(M[2(n+1)-1]) \\ \Sigma^\star &= \underline{M^\star[1]} = M[2(n+1)-1] \end{aligned}$$

Therefore, the tag TE' of the received ciphertext will be:

$$\begin{aligned} TE' &= E(\Sigma^\star \oplus 3^2 L) \\ &= E(M[2(n+1)-1] \oplus 3^2 L) \\ &= E(M[2(n+1)-1] \oplus 2^n L) \\ &= C[2(n+1)-1] \oplus M[2(n+1)] \\ &= TE^\star \end{aligned}$$

Thus, the forged pair $(C^\star, TE^\star)$ will be considered as a valid message. This clearly demonstrates the integrity assurance mechanism is flawed.

6 Proposed Solution

In Sect. 5 we demonstrated that, for certain forms of primitive polynomial, collisions occur between masking values which can be exploited in forgery attacks. This implies that the current choice of masking coefficients cannot be used in a generic construction of OTR. For every choice of finite field the distinctness of the masking values must be verified to ensure the design is secure against forgery attacks.

In this section we propose two minor modifications to OTR which guarantee that the masking coefficients are distinct for any choice of finite field. This makes the generic OTR scheme more robust since it reduces the chance of security compromise as a result of incorrect user choices. Our modifications preserve the main features of OTR mode and still use the powerful doubling masking method.

Note from Table 1 that OTR uses one of four special masks in generating the authentication tag TE from the checksum Σ of the plaintext blocks, with the

choice of mask depending on two message features: whether the number of blocks m is even or odd; and whether the last block is a full block ($|M[m]| = n$) or not. To provide resistance against forgery attacks, it is important that when the multipliers of $2^{l-1}L$ in these masks are considered as powers of 2, the differences between the indexes of any pair of multipliers must be much greater than the maximum possible length (number of blocks) of any plaintext message. This will prevent an attacker forcing collisions between masks by changing the message length (inserting or deleting blocks). We suggest the following design changes to avoid collisions between masks without having to find multipliers at such large distances from one another.

PROPOSED INSTANTIATION OF ENCRYPTION/DECRYPTION CORE We propose two minor modifications, as shown in Table 2 and Fig. 2, to provide a generic version of OTR. Firstly, we set the masking values for odd blocks to start from 2^3L, the masking values for even blocks to start from $2^{-3}L$ and define the four masks to be XOR-ed with the checksum of plaintext blocks as follows:

- 2^2L when m is even and $|M[m]| \neq n$
- $2L$ when m is even and $|M[m]| = n$
- $2^{-2}L$ when m is odd and $|M[m]| \neq n$
- $2^{-1}L$ when m is odd and $|M[m]| = n$

These choices ensure that the masks will not collide regardless of the primitive polynomial being used. The values $2^{-1}L$ and $2^{-2}L$ can easily be obtained from L with the right shift operation instead of the left shift used in the current scheme.

Secondly, we slightly redesign the last step in the process used to compute the tag, separating the XOR of checksum Σ and the number of chunks l in the plaintext message. This will prevent an attacker exploiting the tag computation by compensating between these two variables. In our proposal, changing either variable Σ or l will not have a clear effect on the other. The cost of this change is one extra block cipher call. However, this extra block cipher call makes the generic OTR design more robust.

PROPOSED INSTANTIATION OF AUTHENTICATION CORE As noted is Sect. 3, the authentication core can process the associated data in two modes: serial or parallel. For serial associated data, the same design will be used as it uses only two masks: $2Q$ and 2^2Q. These masks are distinct regardless of the primitive polynomial used. For parallel associated data, we only change the masks used with the last block $A[a]$. OTR (Fig. 1) uses the masks $2^{a-1}3L$ and $2^{a-1}3^2L$ when $|A[a]| \neq n$ and $|A[a]| = n$ respectively. However, there is no need for these two masks to be very far away from each other, as changing the number of blocks a in the associated data will directly affect the accumulated tag TA. Thus, the two masks we suggest for the last block are $2^{-1}L$ and $2^{-2}L$ when $|A[a]| \neq n$ and $|A[a]| = n$ respectively.

Note that the base of all new masks suggested for OTR is 2. This guarantees that the masks will not overlap, and enables us to use the same masks for all choices of n and all choices of the primitive polynomials used to define the finite field $\mathbb{F}_{2^n}$.

Table 2. Proposed OTR algorithm.

Algorithm 5: OTR Encryption	**Algorithm 6**: OTR Decryption
1. $\Sigma \leftarrow 0^n$	1. $\Sigma \leftarrow 0^n$
2. $L \leftarrow E(\underline{N})$	2. $L \leftarrow E(\underline{N})$
3. $(M[1], \ldots, M[m]) \xleftarrow{n} M,\ l = \lceil m/2 \rceil$	3. $(C[1], \ldots, C[m]) \xleftarrow{n} C,\ l = \lceil m/2 \rceil$
4. **for** $i = 1$ **to** $l - 1$ **do**	4. **for** $i = 1$ **to** $l - 1$ **do**
5. $C[2i-1] \leftarrow E(2^{i+2}L \oplus M[2i-1]) \oplus M[2i]$	5. $M[2i-1] \leftarrow E(2^{-(i+2)}L \oplus C[2i-1]) \oplus C[2i]$
6. $C[2i] \leftarrow E(2^{-(i+2)}L \oplus C[2i-1]) \oplus M[2i-1]$	6. $M[2i] \leftarrow E(2^{i+2}L \oplus M[2i-1]) \oplus C[2i-1]$
7. $\Sigma \leftarrow \Sigma \oplus M[2i]$	7. $\Sigma \leftarrow \Sigma \oplus M[2i]$
8. **if** m **is even**	8. **if** m **is even**
9. $Z \leftarrow E(2^{l+2}L \oplus M[m-1])$	9. $M[2m-1] \leftarrow E(2^{-(l+2)}L \oplus \underline{C[m]}) \oplus C[m-1]$
10. $C[m] \leftarrow \mathrm{msb}_{\lvert M[m]\rvert}(Z) \oplus M[m]$	10. $Z \leftarrow E(2^{l+2}L \oplus M[m-1])$
11. $C[2m-1] \leftarrow E(2^{-(l+2)}L \oplus \underline{C[m]}) \oplus M[m-1]$	11. $M[m] \leftarrow \mathrm{msb}_{\lvert M[m]\rvert}(Z) \oplus C[m]$
12. $\Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$	12. $\Sigma \leftarrow \Sigma \oplus Z \oplus \underline{C[m]}$
13. **if** m **is odd**	13. **if** m **is odd**
14. $C[m] \leftarrow \mathrm{msb}_{\lvert M[m]\rvert}(E(2^{l+2}L)) \oplus M[m]$	14. $M[m] \leftarrow \mathrm{msb}_{\lvert M[m]\rvert}(E(2^{l+2}L)) \oplus C[m]$
15. $\Sigma \leftarrow \Sigma \oplus \underline{M[m]}$	15. $\Sigma \leftarrow \Sigma \oplus \underline{M[m]}$
16. **if** m **is even and** $\lvert M[m]\rvert \neq n$	16. **if** m **is even and** $\lvert M[m]\rvert \neq n$
17. $W \leftarrow E(2L \oplus \Sigma)$	17. $W \leftarrow E(2L \oplus \Sigma)$
18. **if** m **is even and** $\lvert M[m]\rvert = n$	18. **if** m **is even and** $\lvert M[m]\rvert = n$
19. $W \leftarrow E(2^2L \oplus \Sigma)$	19. $W \leftarrow E(2^2L \oplus \Sigma)$
20. **if** m **is odd and** $\lvert M[m]\rvert \neq n$	20. **if** m **is odd and** $\lvert M[m]\rvert \neq n$
21. $W \leftarrow E(2^{-1}L \oplus \Sigma)$	21. $W \leftarrow E(2^{-1}L \oplus \Sigma)$
22. **if** m **is odd and** $\lvert M[m]\rvert = n$	22. **if** m **is odd and** $\lvert M[m]\rvert = n$
23. $W \leftarrow E(2^{-2}L \oplus \Sigma)$	23. $W \leftarrow E(2^{-2}L \oplus \Sigma)$
24. $TE \leftarrow E(W \oplus l)$	24. $TE \leftarrow E(W \oplus l)$
25. $C \leftarrow (C[1], \ldots, C[m])$	25. $M \leftarrow (M[1], \ldots, M[m])$
26. **return** (C, TE)	26. **return** (M, TE)
Algorithm 7: OTR Authentication with Parallel A	**Algorithm 8**: OTR Authentication with Serial A
1. $\Xi \leftarrow 0^n$	1. $\Xi \leftarrow 0^n$
2. $Q \leftarrow E(0)$	2. $Q \leftarrow E(0)$
3. $(A[1], \ldots, A[a]) \xleftarrow{n} A$	3. $(A[1], \ldots, A[a]) \xleftarrow{n} A$
4. **for** $i = 1$ **to** $a - 1$ **do**	4. **for** $i = 1$ **to** $l - 1$ **do**
5. $\Xi \leftarrow \Xi \oplus E(Q \oplus A[i])$	5. $\Xi \leftarrow E(\Xi \oplus A[i])$
6. $Q \leftarrow 2Q$	
7. $\Xi \leftarrow \Xi \oplus \underline{A[a]}$	6. $\Xi \leftarrow \Xi \oplus \underline{A[a]}$
8. **if** $\lvert M[m]\rvert \neq n$ **then** $TA \leftarrow E(2^{-1}Q \oplus \Xi)$	7. **if** $\lvert M[m]\rvert \neq n$ **then** $TA \leftarrow E(2Q \oplus \Xi)$
9. **else** $TA \leftarrow E(2^{-2}Q \oplus \Xi)$	8. **else** $TA \leftarrow E(4Q \oplus \Xi)$
10. **return** TA	9. **return** TA

7 Security Bounds for OTR Using the New Masking Coefficients

This section discusses the impact of changing the input masks on the security bounds of OTR. Firstly, all of proposed masks for OTR have the form 2^j. Since 2 is the generator of the finite field $\mathbb{F}_{2^n}$ and the order of this field is $2^n - 1$, this assures that $2^j = 2^{j'}$ iff $j = j'$ for any $j \leq 2^n - 1$. Therefore, these masks will not collide for any collection of blocks with a length less than $(2^{n-1} - 3)$, which is well beyond the message length restriction of $2^{n/2}$ blocks imposed by the designer of OTR. Thus, the collision attacks discussed in Sect. 5 are now precluded.

Secondly, the security proofs of OTR [9] assume that all tweakable block cipher calls in each of the two rounds have distinct masks. As shown in the

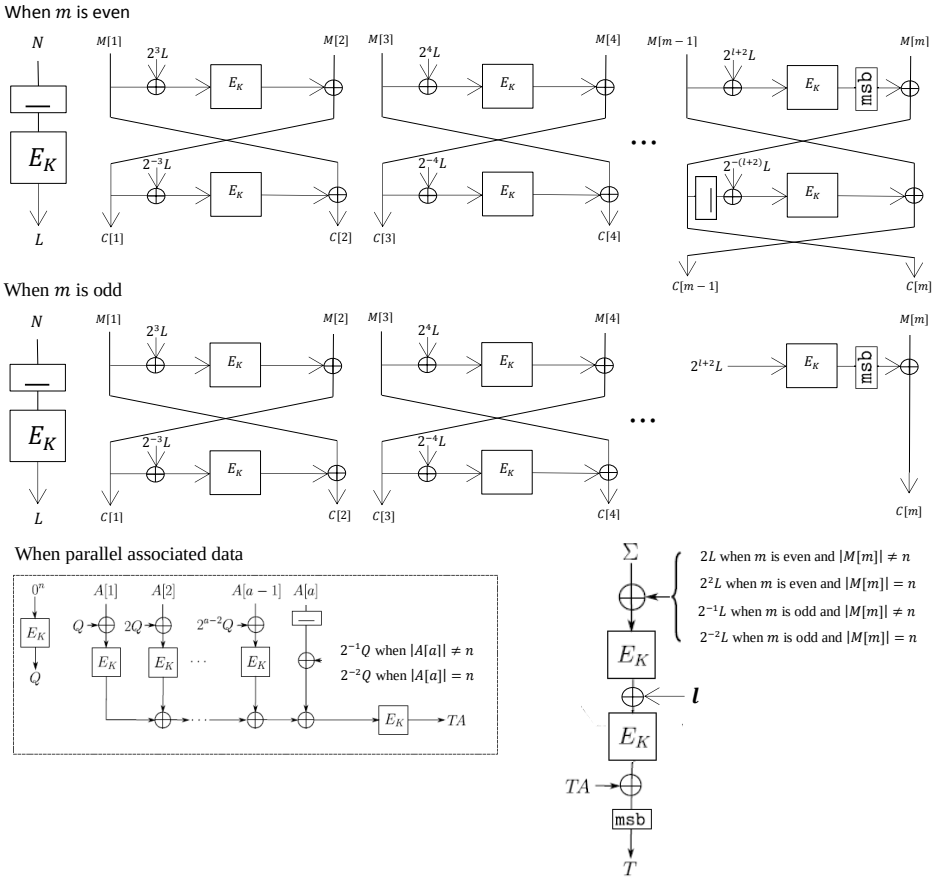

Fig. 2. Proposed OTR encryption diagram with parallel associated data.

above paragraph, the proposed masks are guaranteed to be different from each other; thus, the same security bounds for OTR still hold.

Finally, our proposed modification to OTR adds one extra block cipher call, as shown in Fig. 2. This step is required in order to avoid using a mask with a base other than 2. The probability that an attacker can guess the tag successfully is still $1/2^\tau$ where is τ the tag length. Therefore, the security of OTR will not be degraded with the new instantiation method.

8 Conclusion

OTR is a block cipher mode of operation for AEAD that uses a doubling masking technique. OTR is designed to be applicable for any block size n but currently requires a suitable choice to be made for the finite field $\mathbb{F}_{2^n}$ used for doubling the mask values. The security of OTR against forgery attacks depends on the

distinctness of the masking values. This has only been proved for specific choices of primitive polynomial in the particular cases of $n = 64$ and $n = 128$.

In this paper, we show that the masks used in the current instantiation of OTR are not distinct for certain choices of finite field. Using these choices, we demonstrate practical forgery attacks against OTR. Thus, the generic form of the OTR design is not secure.

We propose two minor modifications to OTR to make the generic version of this scheme more robust. We do this by specifying a set of masks that are distinct in every finite field $\mathbb{F}_{2^n}$. This enables OTR to work with any finite field without invalidating the security claimed. Note that this work does not imply that the versions of OTR described in [8, 9] are insecure.

9 Acknowledgements

Hassan Al Mahri would like to acknowledge the scholarship for this research from the government of the Sultanate of Oman.

References

1. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: AES-COPA, `http://competitions.cr.yp.to/caesar-submissions.html` (2014)
2. Bernstein, D.: Cryptographic Competitions: CAESAR, `http://competitions.cr.yp.to/caesar-call.html` (2014)
3. Bost, R., Sanders, O.: Trick or Tweak: On the (In)security of OTR's Tweaks. Cryptology ePrint Archive, Report 2016/234, `http://eprint.iacr.org/` (2016)
4. Datta, N., Nandi, M.: ElmD, `http://competitions.cr.yp.to/caesar-submissions.html` (2014)
5. Halevi, S., Rogaway, P.: A Parallelizable Enciphering Mode. In: Topics in Cryptology–CT-RSA 2004, pp. 292–304. Springer (2004)
6. Iwata, T., Kurosawa, K.: OMAC: One-key CBC MAC. In: International Workshop on Fast Software Encryption, pp. 129–153. Springer (2003)
7. Liskov, M., Rivest, R., Wagner, D.: Tweakable Block Ciphers. In: Advances in Cryptology–CRYPTO 2002, pp. 31–46. Springer (2002)
8. Minematsu, K.: AES-OTR, `http://competitions.cr.yp.to/caesar-submissions.html` (2014)
9. Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. Cryptology ePrint Archive, Report 2013/628, `http://eprint.iacr.org/` (2013)
10. Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In: Advances in Cryptology–EUROCRYPT 2014, pp. 275–292. Springer (2014)
11. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: Advances in Cryptology–ASIACRYPT 2004, pp. 16–31. Springer (2004)
12. Seroussi, G.: Table of Low-Weight Binary Irreducible Polynomials. Computer Systems Laboratory, Report HPL-98-135, `http://www.hpl.hp.com/techreports/98/HPL-98-135.pdf?jumpid=reg_R1002_USEN` (1998)